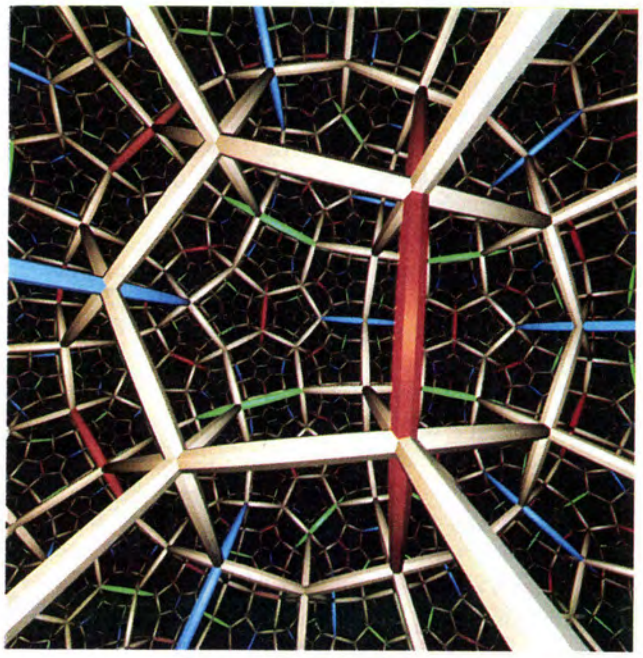


Figure 5: A view of the tessellation of hyperbolic space by regular right-angled dodecahedra, as in the movie "Not Knot". This image was rendered using Renderman.

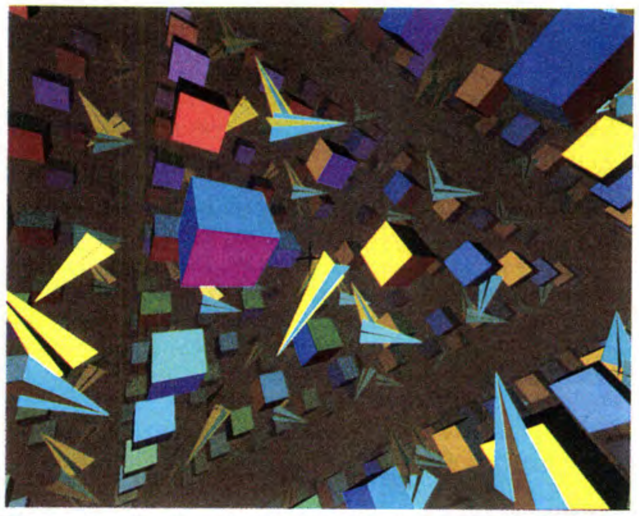


Figure 7: A view inside the Euclidean orbifold from "Not Knot" with the camera as a paper airplane.

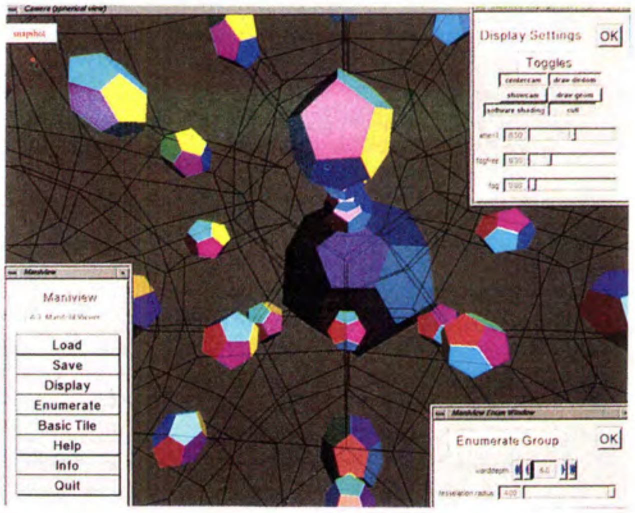


Figure 8: A typical session of **Maniview** showing some of its panels but hiding **Geomview** panels.

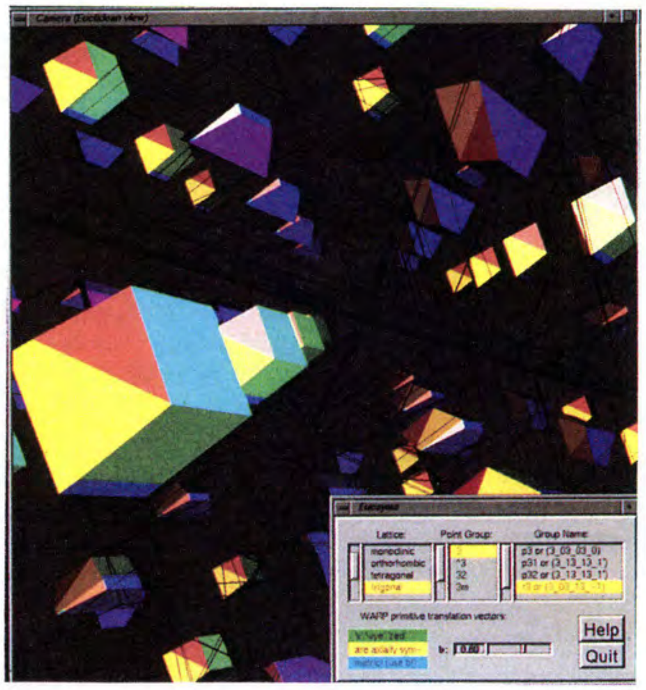


Figure 6: A snapshot of a session using **eucsyms**, an application for exploring the 230 Euclidean crystal groups.

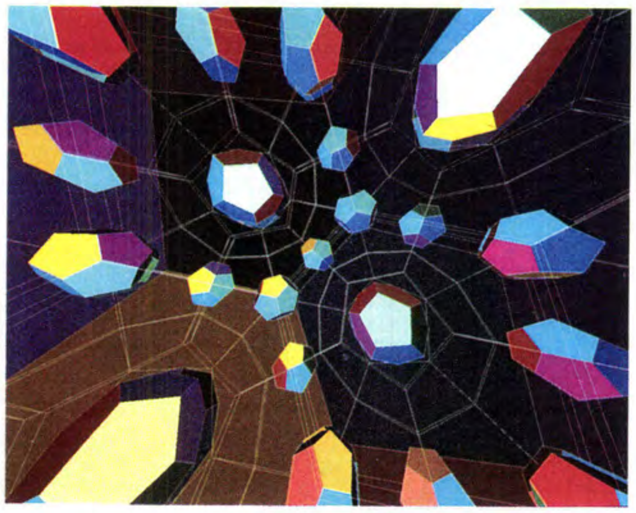


Figure 9: A view of the 120-cell. We reduce the size of the shaded polyhedra but draw the original edges.

of a cylinder in these spaces? Also, neither space allows similarity transformations: changing the size of an object changes its shape! Other questions arise. What sort of harmonic analysis is available to synthesize fractal terrains and textures in these spaces? If we hope to do physically-based modeling in these spaces, we need to expand our understanding of the laws of physics beyond the behavior of light described above in relation to shading. Finally, the theory of splines in non-Euclidean spaces was explored in [GK85].

In the area of topological content, one obvious goal is to implement the five non-isotropic three dimensional model geometries. Also, there are many sorts of discrete groups, particularly those that create fractal patterns, which do not fit neatly into the current framework.

In the direction of mathematical research and user interface, the efforts described here suggest various techniques for exploring 3-manifolds. Connecting this software with virtual reality technology would allow the researcher to perform a variety of explorations of the space. The use of sound also promises to yield useful evidence.

Looking at the wider world of Riemannian geometry, this work is one step in the direction of visualizing arbitrary curved spaces, the Riemannian manifolds that figure centrally in relativity and cosmology. For related work see [HD89].

Finally, this work opens a new domain for artistic creativity, three dimensional analogues of M. C. Escher's dramatic interlocking planar tessellations.

9 Conclusions

Approaching metric geometries via their common ancestry in projective geometry yields simple models which can be directly implemented in existing rendering systems. The resulting systems allow interactive navigation of curved spaces for the first time. Custom shaders provide realistic rendering of the insider's view. Methods for manipulating and displaying discrete groups allow interactive exploration of a wide class of topological manifolds modeled on these spaces, that have never been visualized before. The resulting system provides a unique tool for mathematicians, educators, and scientists and artists whose work is related to spatial symmetry.

Acknowledgements

I would like to acknowledge valuable ideas, comments, suggestions, and assistance from Bill Thurston, Jeff Weeks, Silvio Levy, Stuart Levy, Mark Phillips, John Sullivan, David Banks, and the reviewers.

References

[Bea83] Alan F. Beardon. *The Geometry of Discrete Groups.* Springer-Verlag, 1983.

[Car76] Manfredo P. Do Carmo. *Differential Geometry of Curves and Surfaces.* Prentice-Hall, 1976.

[Cay59] A. Cayley. A sixth memoir upon quantics. *Philosophical Transactions of the Royal Society of London*, 149:61–90, 1859.

[Cox65] H.M.S. Coxeter. *Non-Euclidean Geometry.* University of Toronto Press, 1965.

[Cox73] H.M.S. Coxeter. *Regular Polytopes.* Dover, 1973.

[Cox87] H.M.S. Coxeter. *Projective Geometry.* Springer Verlag, 1987.

[ECH+91] D. B. A. Epstein, Jim Cannon, Derek Holt, Silvio Levy, Mike Patterson, and William Thurston. *Word Processing in Groups.* Jones and Bartlett, 1991.

[FRC92] Helaman Ferguson, Alyn Rockwood, and Jordan Cox. Topological design of sculptural surfaces. *Computer Graphics*, 26:149–156, July, 1992. Proceedings of SIGGRAPH 1992.

[GK85] S. Gabriel and J. Kajiya. Spline interpolation in curved space. In *State of the Art Image Synthesis*, 1985. Course notes for SIGGRAPH 1985.

[GM91] Charlie Gunn and Delle Maxwell. *Not Knot.* Jones and Bartlett, 1991.

[Gun83] Charlie Gunn. A computer implementation of the two-dimensional euclidean crystallographic groups. Master's thesis, UNC, Chapel Hill, 1983.

[Gun92] Charlie Gunn. Visualizing hyperbolic geometry. In *Computer Graphics and Mathematics*, pages 299–313. Eurographics, Springer Verlag, 1992.

[HD89] Ping-Kang Hsiung and Robert H.P. Dunn. Visualizing relativistic effects in spacetime. In *Supercomputing 89.* IEEE/ACM, Nov, 1989.

[Lev92] Silvio Levy. Automatic generation of hyperbolic tilings. *Leonardo*, 35:349–354, 1992.

[LM78] E. H. Lockwood and R. H. Macmillan. *Geometric symmetry.* Cambridge University Press, 1978.

[MLP+] Tamara Munzner, Stuart Levy, Mark Phillips, Nathaniel Thurston, and Celeste Fowler. Geomview — an interactive viewing program for sgi workstations. ftp@geom.umn.edu.

[Mun75] James Munkres. *Topology: A First Course*, chapter 8. Prentice-Hall, 1975.

[PG92] Mark Phillips and Charlie Gunn. Visualizing hyperbolic space: Unusual uses of 4x4 matrices. In *1992 Symposium on Interactive 3D Graphics*, pages 209–214. ACM SIGGRAPH, ACM, 1992.

[Sch80] R.L.E. Schwarzenberger. *N-Dimensional Crystallography.* Pitman Publishing, 1980. chapters 13-16.

[Thu82] William Thurston. Three dimensional manifolds, kleinian groups and hyperbolic geometry. *BAMS*, 19:417–431, 1982.

[Thuar] William Thurston. *The Geometry and Topology of Three-Manifolds*, volume 1. to appear.

[Ups89] Steve Upstill. *The Renderman Companion.* Addison-Wesley, 1989. chapters 13-16.

[Wee] Jeff Weeks. snappea — a macintosh application for computing 3-manifolds. ftp@geom.umn.edu.

[Wee85] Jeff Weeks. *The Shape of Space.* Marcel Dekker, 1985.

[Woo22] Frederick Woods. *Higher Geometry.* Dover, 1961 (1922).

Imaging Vector Fields Using Line Integral Convolution

Brian Cabral
*Leith (Casey) Leedom**

Lawrence Livermore National Laboratory

ABSTRACT

Imaging vector fields has applications in science, art, image processing and special effects. An effective new approach is to use linear and curvilinear filtering techniques to locally blur textures along a vector field. This approach builds on several previous texture generation and filtering techniques[8, 9, 11, 14, 15, 17, 23]. It is, however, unique because it is local, one-dimensional and independent of any predefined geometry or texture. The technique is general and capable of imaging arbitrary two- and three-dimensional vector fields. The local one-dimensional nature of the algorithm lends itself to highly parallel and efficient implementations. Furthermore, the curvilinear filter is capable of rendering detail on very intricate vector fields. Combining this technique with other rendering and image processing techniques — like periodic motion filtering — results in richly informative and striking images. The technique can also produce novel special effects.

CR categories and subject descriptors: I.3.3 [Computer Graphics]: Picture/Image generation; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism; I.4.3 [Image Processing]: Enhancement.

Keywords: convolution, filtering, rendering, visualization, texture synthesis, flow fields, special effects, periodic motion filtering.

1. INTRODUCTION

Upon first inspection, imaging vector fields appears to have limited application — confined primarily to scientific visualization. However, much of the form and shape in our environment is a function of not only image intensity and color, but also of directional information such as edges. Painters, sculptors, photographers, image processors[16] and computer graphics researchers[9] have recognized the importance of direction in the process of image creation and form. Hence, algorithms that can image such directional information have wide application across both scientific and artistic domains.

Such algorithms should possess a number of desirable and sometimes conflicting properties including: accuracy, locality of calculation, simplicity, controllability and generality. Line Integral Convolution (LIC) is a new technique that possesses many of these properties. Its generality allows for the introduction of a completely new family of periodic motion filters which have wide application (see section 4.1). It represents a confluence of signal and image processing and a variety of previous work done in computer graphics and scientific visualization.

* Authors' current e-mail addresses are: *cabral@llnl.gov* and *casey@gauss.llnl.gov.*

2. BACKGROUND

There are currently few techniques which image vector fields in a general manner. These techniques can be quite effective for visualizing vector data. However, they break down when operating on very dense fields and do not generalize to other applications. In particular, large vector fields (512x512 or greater) strain existing algorithms.

Most vector visualization algorithms use spatial resolution to represent the vector field. These include sampling the field, such as with stream lines[12] or particle traces, and using icons[19] at every vector field coordinate. Stream lines and particle tracing techniques depend critically on the placement of the "streamers" or the particle sources. Depending on their placement, eddies or currents in the data field can be missed. Icons, on the other hand, do not miss data, but use up a considerable amount of spatial resolution limiting their usefulness to small vector fields.

Another general approach is to generate textures via a vector field. Van Wijk's *spot noise* algorithm[23] uses a vector field to control the generation of bandlimited noise. The time complexity of the two types of implementation techniques presented by Van Wijk are relatively high. Furthermore the technique, by definition, depends heavily on the form of the texture (spot noise) itself. Specifically, it does not easily generalize to other forms of textures that might be better suited to a particular class of vector data (such as fluid flow versus electromagnetic).

Reaction diffusion techniques[20, 24] also provide an avenue for visualizing vector fields since the controlling differential equations are inherently vector in nature. It is possible to map vector data onto these differential equations to come up with a vector visualization technique. Here too however, the time complexity of these algorithms limit their general usefulness.

Three-dimensional vector fields can be visualized by three-dimensional texture generation techniques such as texels and hypertextures described in [11, 15]. Both techniques take a texture on a geometrically defined surface and project the texture out some distance from the surface. By definition these techniques are bound to the surface and do not compute an image for the entire field as is done by Van Wijk[23]. This is limiting in that it requires a priori knowledge to place the surface. Like particle streams and vector streamers these visualization techniques are critically dependent on the placement of the sampling surface.

The technique presented by Haeberli[9] for algorithmicly generating "paintings" via vector-like brush strokes can also be thought of as a vector visualization technique. Crawfis and Max[5] describe a three-dimensional variation on this in which blurred cylinders represent three-dimensional brush strokes whose directions and colors are controlled by a three-dimensional vector field. Both techniques represent a conceptual extension of traditional icon placement, where the icons are more sophisticated shapes. However, these techniques break down as the density of the field increases since they require spatial resolution to work.

What is needed is a technique that can image dense vector fields, is independent of both predefined sampling placement constraints and texture generation techniques and can work in two and three dimensions. Such a technique would be very general and have wide application.

3. DDA CONVOLUTION

One approach is a generalization of traditional DDA line drawing techniques[1] and the spatial convolution algorithms described by Van Wijk[23] and Perlin[14]. Each vector in a field is used to define a long, narrow, DDA generated filter kernel tangential to the vector and going in the positive and negative vector direction some fixed distance, *L*. A texture is then mapped one-to-one onto the vector field. The input texture pixels under the filter kernel are summed, normalized by the length of the filter kernel, *2L*, and placed in an output pixel image for the vector position. Figure 1, illustrates this operation for a single vector in a field.

This effectively filters the underlying texture as a function of the vector field. The images in figure 2 are rendered using the DDA convolution algorithm. On the left is a simple circular vector field; to its right is the result of a computational fluid dynamics code. The input texture image in these examples is white noise. Although the description above implies a box filter, any arbitrary filter shape can be used for the filter convolution kernel. It is important to note that this algorithm is very sensitive to symmetry of the DDA algorithm and filter. If the algorithm weights the forward direction more than the backward direction, the circular field in figure 2 appears to spiral inward implying a vortical behavior that is not present in the vector field.

3.1 LOCAL FIELD BEHAVIOR

The DDA approach, while efficient, is inherently inaccurate. It assumes that the local vector field can be approximated by a straight line. For points in vector fields where the local radius of curvature is large, this assumption is valid. However, where there are complex structures smaller than the length of the DDA line, the local radius of curvature is small and is not well approximated by a straight line. In a sense, DDA convolution renders the vector field unevenly, treating linear portions of the vector field more accurately than small scale vortices. While this graceful degradation may be fine or even desirable for special effects applications, it is problematic for visualizing vector fields such as the ones in figure 2, since detail in the small scale structures is lost.

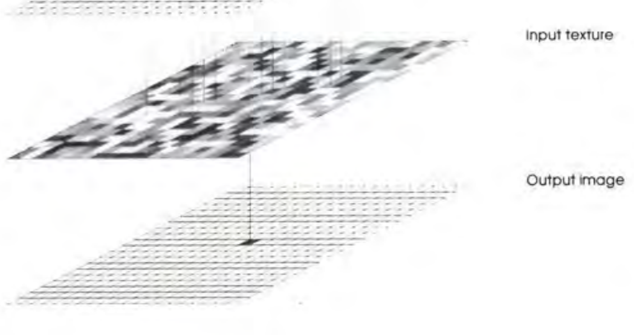


Figure 1: The mapping of a vector onto a DDA line and input pixel field generating a single output pixel.

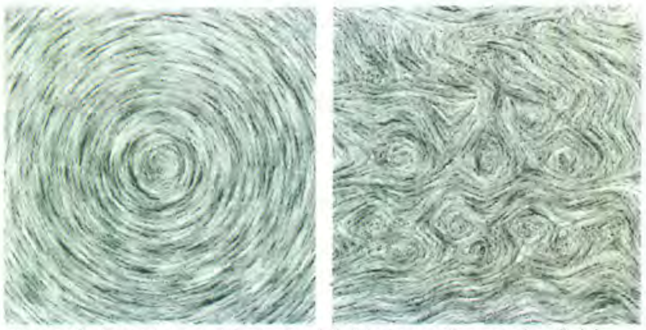


Figure 2: Circular and turbulent fluid dynamics vector fields imaged using DDA convolution over white noise.

Van Wijk's spot noise algorithm[23] also suffers from this problem since the spots are elliptically stretched along a line in the direction of the local field. If the ellipse major axis exceeds the local length scale of the vector field, the spot noise will inaccurately represent the vector field. An accurate measure of local field behavior would require a global analysis of the field. Such techniques currently do not exist for arbitrary vector fields, would most likely be expensive to calculate[13] and are an area of active research[7].

4. LINE INTEGRAL CONVOLUTION

The local behavior of the vector field can be approximated by computing a local stream line that starts at the center of pixel (x, y) and moves out in the positive and negative directions.[1] The forward coordinate advection is given by equation (1).

$$P_0 = (x + 0.5, y + 0.5)$$

$$P_i = P_{i-1} + \frac{V(\lfloor P_{i-1} \rfloor)}{\| V(\lfloor P_{i-1} \rfloor) \|} \Delta s_{i-1} \tag{1}$$

$V(\lfloor P \rfloor)$ = the vector from the input vector field at lattice point $(\lfloor P_x \rfloor, \lfloor P_y \rfloor)$

$$s_e = \begin{cases} \infty \text{ if } V \parallel e \\ 0 \text{ if } \dfrac{\lfloor P_c \rfloor - P_c}{V_c} < 0 \\ \dfrac{\lfloor P_c \rfloor - P_c}{V_c} \text{ otherwise} \end{cases} \quad \text{for } (e, c) \in \begin{Bmatrix} (top, y) \\ (bottom, y) \\ (left, x) \\ (right, x) \end{Bmatrix} \tag{2}$$

$$\Delta s_i = \min(s_{top}, s_{bottom}, s_{left}, s_{right})$$

[1] Vector field lattice and image coordinates are usually specified in a left-handed coordinate system while vector components are usually specified in a right-handed coordinate system. In this case, the *y*-component of the lattice coordinate in equation (1) must be reflected about the vertical center of the lattice to operate in a consistent coordinate system. This reflection has been omitted to preserve simplicity of presentation.

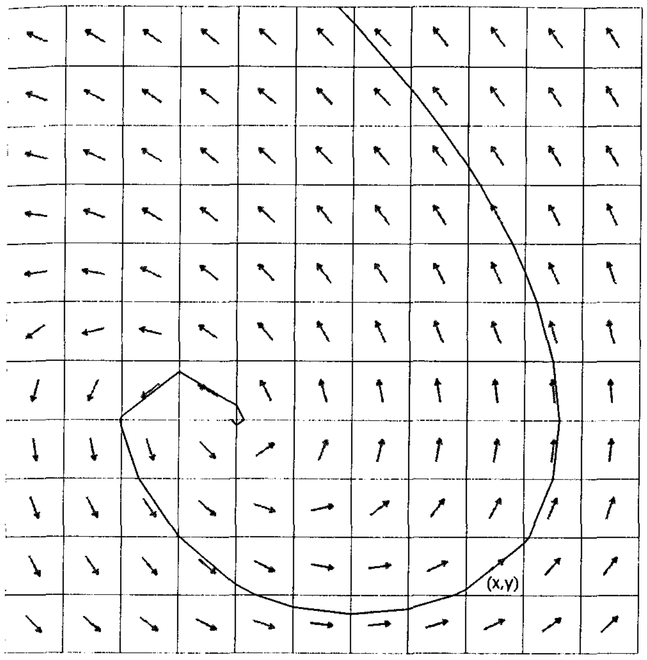


Figure 3: A two-dimensional vector field showing the local stream line starting in cell (*x, y*). The vector field is the upper left corner of the fluid dynamics field in figures 2 and 4.

Only the directional component of the vector field is used in this advection. The magnitude of the vector field can be used later in post processing steps as explained in section 4.3.1. Δs_i is the positive parametric distance along a line parallel to the vector field from P_i to the nearest cell edge.

As with the DDA algorithm, it is important to maintain symmetry about a cell. Hence, the local stream line is also advected backwards by the negative of the vector field as shown in equation (3).

$$P'_0 = P_0$$

$$P'_i = P'_{i-1} - \frac{V(\lfloor P'_{i-1} \rfloor)}{\| V(\lfloor P'_{i-1} \rfloor) \|} \Delta s'_{i-1} \qquad (3)$$

Primed variables represent the negative direction counterparts to the positive direction variables and are not repeated in subsequent definitions. As above $\Delta s'_i$, is always positive.

The calculation of Δs_i in the stream line advection is sensitive to round off errors. Δs_i must produce advected coordinates that lie within the i+1th cell, taking the stream line segment out of the current cell. In the implementation of the algorithm a small round off term is added to each Δs_i to insure that entry into the adjacent cell occurs. This local stream line calculation is illustrated in figure 3. Each cell is assumed to be a unit square. All spatial quantities (e.g., Δs_i) are relative to this measurement. However, the cells need not be square or even rectangular (see section 6) for this approximation to work. So, without loss of generality, descriptions are given relative to a cubic lattice with unit spacing.

Continuous sections of the local stream line — i.e. the straight line segments in figure 3 — can be thought of as parameterized space curves in *s* and the input texture pixel mapped to a cell can be treated as a continuous scalar function of x and y.[2] It is then possible to integrate over this scalar field along each parameterized space curve. Such integrals can be summed in a piecewise $\mathbf{C}^1$ fashion and are known as line integrals of the first kind (LIFK)[2]. The convolution concept used in the DDA algorithm can now be combined with LIFK to form a Line Integral Convolution (LIC). This results in a variation of the DDA approach that locally follows the vector field and captures small radius of curvature features. For each continuous segment, *i*, an exact integral of a convolution kernel *k*(*w*) is computed and used as a weight in the LIC as shown in equation (4).

$$h_i = \int_{s_i}^{s_i + \Delta s_i} k(w)\, dw \qquad (4)$$

where

$$s_0 = 0$$

$$s_i = s_{i-1} + \Delta s_{i-1}$$

The entire LIC for output pixel *F'(x, y)* is given by equation (5).

$$F'(x, y) = \frac{\sum_{i=0}^{l} F(\lfloor P_i \rfloor) h_i + \sum_{i=0}^{l'} F(\lfloor P'_i \rfloor) h'_i}{\sum_{i=0}^{l} h_i + \sum_{i=0}^{l'} h'_i} \qquad (5)$$

where

$F(\lfloor P \rfloor)$ is the input pixel corresponding to the vector at position $(\lfloor P_x \rfloor, \lfloor P_y \rfloor)$

$$l = i \text{ such that } s_i \le L < s_{i+1} \qquad (6)$$

The numerator of equation (5) represents the line integral of the filter kernel times the input pixel field, *F*. The denominator is the line integral of the convolution kernel and is used to normalize the output pixel weight (see section 4.2).

The length of the local stream line, *2L*, is given in unit pixels. Depending on the input pixel field, *F*, if *L* is too large, all the resulting LICs will return values very close together for all coordinates (*x, y*). On the other hand, if *L* is too small then an insufficient amount of filtering occurs. Since the value of *L* dramatically affects the performance of the algorithm, the smallest effective value is desired. For most of the figures, a value of 10 was used.

Singularities in the vector field occur when vectors in two adjacent local stream line cells geometrically "point" at a shared cell edge. This results in Δs_i values equal to zero leaving *l* in equation (6) undefined. This situation can easily be detected and the advection algorithm terminated. If the vector field goes to zero at any point, the LIC algorithm is terminated as in the case of a field singularity. Both of these cases generate truncated stream lines. If a zero field vector lies in the starting cell of the LIC, the input pixel value for that cell, a constant or any other arbitrary value can be returned as the value of the LIC depending on the visual effect desired for null vectors.

Using adjacent stream line vectors to detect singularities can however result in false singularities. False singularities occur when the vector field is nearly parallel to an edge, but causes the LIC to cross over that edge. Similarly, the cell just entered also has a near parallel vector which points to this same shared edge. This artifact can be remedied by adjusting the parallel vector/edge test found in equation (2), to test the angle formed between the vector and the edge against some small angle *theta*, instead of zero. Any vector which forms an angle less than *theta* with some edge is deemed to be "parallel" to that edge. Using a value of 3° for *theta* removes these artifacts.

The images in figure 4 were rendered using LIC and correspond to the same two vector fields rendered in figure 2. Note the increased amount of detail present in these images versus their DDA counterparts. In particular the image of the fluid dynamics vector field in figure 4 shows detail incorrectly rendered or absent in figure 2.

[2] Bilinear, cubic or Bezier splines are viable alternatives to straight line segments. However, these higher order curves are more expensive to compute.

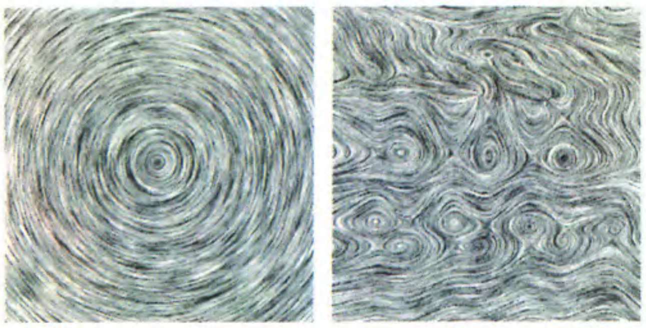


Figure 4: Circular and turbulent fluid dynamics vector fields imaged using LIC over white noise.

The images in figure 5 show the effect of varying *L*. The input texture is a photograph of flowers. The input vector field was created by taking the gradient of a bandlimited noise image and rotating each of the gradient vectors by 90°, producing vectors which follow the contours of the soft hills and valleys of the bandlimited noise. With *L* equal to 0, the input image is passed through unchanged. As the value of *L* increases, the input image is blurred to a greater extent, giving an impressionistic result. Here, a biased ramp filter[10] is used to roughly simulate a brush stroke.

Figures 2, 4, 8, 9 and 11 were generated using white noise input images. Aliasing can be a serious problem when using LIC with a high frequency source image such as white noise. The aliasing is caused by the one-dimensional point sampling of the infinitely thin LIC filter. This aliasing can be removed by either creating a thick LIC filter with a low-pass filter cross section or by low-pass filtering the input image. This second alternative is preferable since it comes at no additional cost to the LIC algorithm. The images in figure 6 show the effect of running LIC over 256x256 white noise which has been low-pass filtered using a fourth order Butterworth filter with cutoff frequencies of 128, 84, 64, and 32.

It is worth noting that Van Wijk's spot noise algorithm[23] can be adapted to use the local stream line approximation to more accurately represent the behavior of a vector field. Instead of straight line elliptical stretching, each spot could be warped so that the major axis follows the local stream line. Furthermore, the minor axis could either be perpendicular to the warped major axis or itself could be warped along transverse field lines. However, an algorithm to perform this task for an arbitrary local stream line would be inherently more expensive and complex than the LIC algorithm.

Sims[18] describes an alternative technique which produces results similar to LIC. This alternative approach warps or advects texture coordinates as a function of a vector field. The similarity between the two techniques is predictable even though the techniques are quite different. The dilation and contraction of the texture coordinate system warping has the visual effect of blurring and sharpening the warped image. This is due to the resampling and reconstruction process necessary when warping from one coordinate system to another. Thus, for regions where the source image is stretched along the vector field an apparent blurring will occur similar to those seen with LIC. However, the techniques are completely different in two fundamental ways. First, LIC is a local operator, meaning no information outside of a fixed area of interest is needed. Warping even when done locally requires maintaining global consistency to avoid tearing holes in the warped image. This increases the complexity of the warping operation when compared to LIC. Second, LIC is a spatially varying filtering operation and does not warp or transform any texture coordinates.

4.1 PERIODIC MOTION FILTERS

The LIC algorithm visualizes local vector field tangents, but not their direction. Freeman, et al[8] describe a technique which simulates motion by use of special convolutions. A similar technique is used by Van Gelder and Wilhelms[22] to show vector field flow. This technique can be extended and used to represent the local vector field direction via animation of successive LIC imaged vector fields using varying phase shifted periodic filter kernels.

The success of this technique depends on the shape of the filter. In the previous examples (figures 2 and 4), a constant or box filter is used. If the filter is periodic like the filters used in [8], by changing the phase of such filters as a function of time, apparent motion

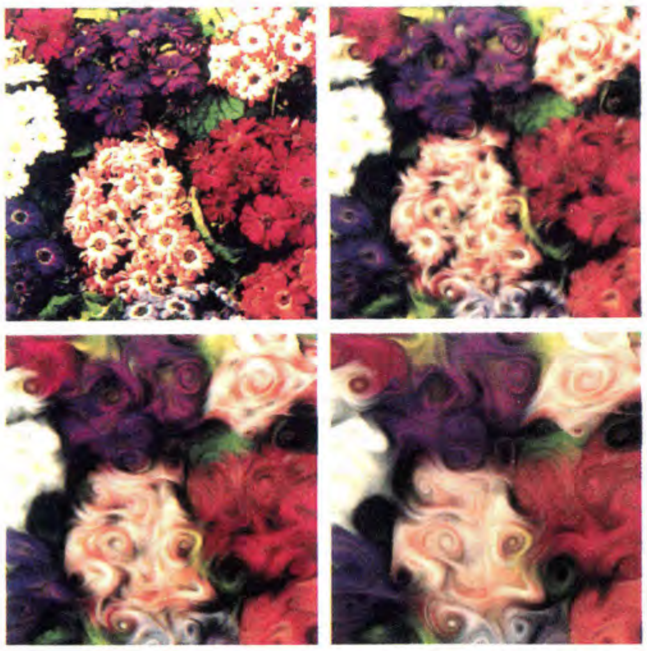


Figure 5: Photograph of flowers processed using LIC with *L* equal to 0, 5, 10 and 20 (left to right, top to bottom).

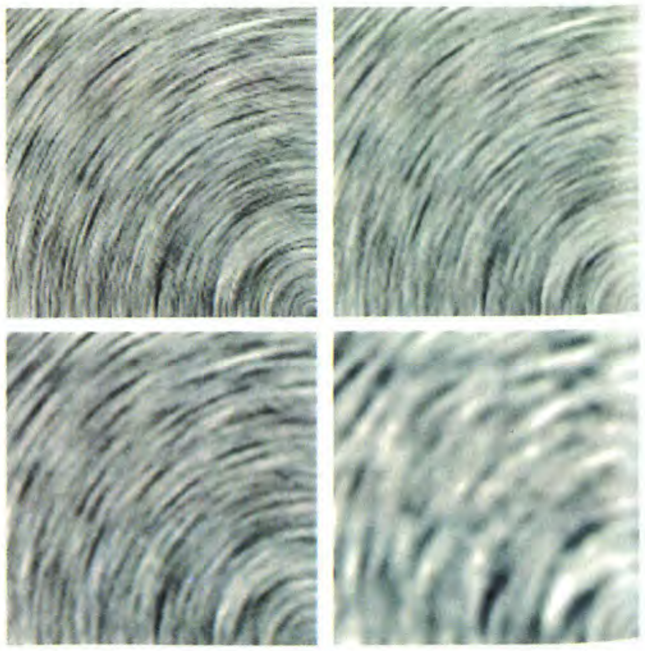


Figure 6: The upper left hand quarter of the circular vector field is convolved using LIC over Butterworth low-pass filtered white noise with cutoff frequencies of 128, 86, 64, and 32 (left to right, top to bottom).

in the direction of the vector field is created. However, the filters used in [8] were, by design, high-pass Laplacian edge enhancing filters. Using this filter over a bandlimited noise texture produces very incoherent images since the high frequency components of the noise are accentuated. Instead, it is possible, and desirable, to create periodic low-pass filters to blur the underlying texture in the direction of the vector field. A Hanning filter, $1/2(1 + \cos(w+\beta))$, has this property. It has low band-pass filter characteristics, it is periodic by definition and has a simple analytic form. This function will be referred to as the *ripple* filter function.

Since the LIC algorithm is by definition a local operation, any filter used must be windowed. That is, it must be made local even if it has infinite extent. In the previous section we used a constant filter implicitly windowed by a box of height one. Using this same box window on a phase shifted Hanning filter we get a filter with abrupt cutoffs, as illustrated in the top row of figure 7.

This abrupt cutoff is noticeable as spatio-temporal artifacts in animations that vary the phase as a function of time. One solution to this problem is to use a Gaussian window as suggested by Gabor[4].[3] By multiplying, or windowing, the Hanning function by a Gaussian, these cutoffs are smoothly attenuated to zero. However, a Gaussian windowed Hanning function does not have a simple closed form integral. An alternative is to find a windowing function with windowing properties similar to a Gaussian and which has a simple closed form integral. Interestingly, the Hanning function itself meets these two criteria. In the bottom row of figure 7, the five phase shifted Hanning filter functions in the top row are multiplied by the Hanning window function in the middle row. The general form of this function is shown in equation (7). In this equation c and d represent the dilation constants of the Hanning window and ripple functions respectively. β is the ripple function phase shift given in radians. The integral of $k(w)$ from a to b used in equation (4) is shown in equation (8).

$$k(w) = \frac{1+\cos(cw)}{2} \times \frac{1+\cos(dw+\beta)}{2} \qquad (7)$$
$$= \frac{1}{4}\left(1+\cos(cw)+\cos(dw+\beta)+\cos(cw)\cos(dw+\beta)\right)$$

$$\int_a^b k(w)\,dw \qquad (8)$$
$$= \frac{1}{4}\left(\begin{array}{c} b-a+\dfrac{\sin(bc)-\sin(ac)}{c} \\ +\dfrac{\sin(bd+\beta)-\sin(ad+\beta)}{d} \\ +\dfrac{\sin(b(c-d)-\beta)-\sin(a(c-d)-\beta)}{2(c-d)} \\ +\dfrac{\sin(b(c+d)+\beta)-\sin(a(c+d)+\beta)}{2(c+d)} \end{array}\right)$$

As mentioned above, both the Hanning window and the Hanning ripple filter function can be independently dilated by adjusting c and d to have specific local support and periodicity. The window function has a fixed period of 2π.

Choosing the periodicity of the ripple function represents making a design trade-off between maintaining a nearly constant frequency response as a function of phase shift and the quality of the

[3]. D. Gabor in 1946 created a localized form of the Fourier transform known as the Gabor transform. This transform is the Fourier transform of an input signal multiplied by a Gaussian window translated along the signal as a function of time. The net result is a signal which is spatially and frequency localized. Wavelet theory is based on a generalization of this type of spatial and frequency localization.

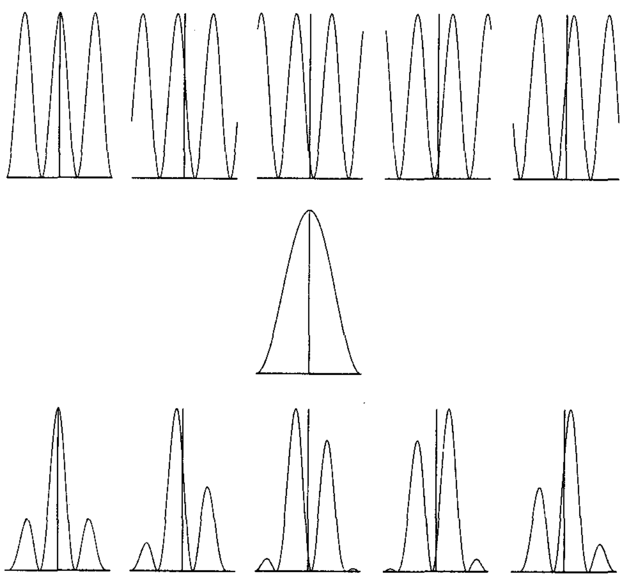


Figure 7: Phase shifted Hanning ripple functions(top), a Hanning windowing function(middle), and Hanning ripple functions multiplied by the Hanning window function(bottom).

apparent motion[3]. A low frequency ripple function results in a windowed filter whose frequency response noticeably changes as a function of phase. This appears as a periodic blurring and sharpening of the image as the phase changes. Higher frequency ripple functions produce windowed filters with a nearly constant frequency response since the general shape of the filter doesn't radically change. However, the feature size picked up by the ripple filter is smaller and the result is less apparent motion. If the ripple frequency exceeds the Nyquist limit of the pixel spacing the apparent motion disappears. Experimentation shows that a ripple function frequency between 2 and 4 cycles per window period is reasonable. One can always achieve both good frequency response and good feature motion by increasing the spatial resolution. This comes, of course, at a cost of increased computation[16].

4.2 NORMALIZATION

A normalization to the convolution integral is performed in equation (5) to insure that the apparent brightness and contrast of the resultant image is well behaved as a function of kernel shape, phase and length. The numerator in equation (5) is divided by the integral of the convolution kernel. This insures that the normalized area under the convolution kernel is always unity resulting in a constant overall brightness for the image independent of the filter shape and LIC length.

Because the actual length of the LIC may vary from pixel to pixel, the denominator can not be precomputed. However, an interesting effect is observed if a fixed normalization is used. Truncated stream lines are attenuated which highlights singularities. The images in figure 8 a show another section of the fluid dynamics vector field imaged with variable and constant kernel normalization. The implementation of the LIC algorithm uses precomputed sum tables for the integral to avoid costly arithmetic in the innermost loop.

A second normalization may be done to insure the output image retains the input image's contrast properties. The LIC algorithm reduces the overall image contrast as a function of L. In fact, in the case of the box filter, as L goes to infinity the entire output image goes to the average of the input image. This can be ameliorated by amplifying the input or contrast stretching the output image as a function of L. Clearly as L goes to infinity the amplification or con-

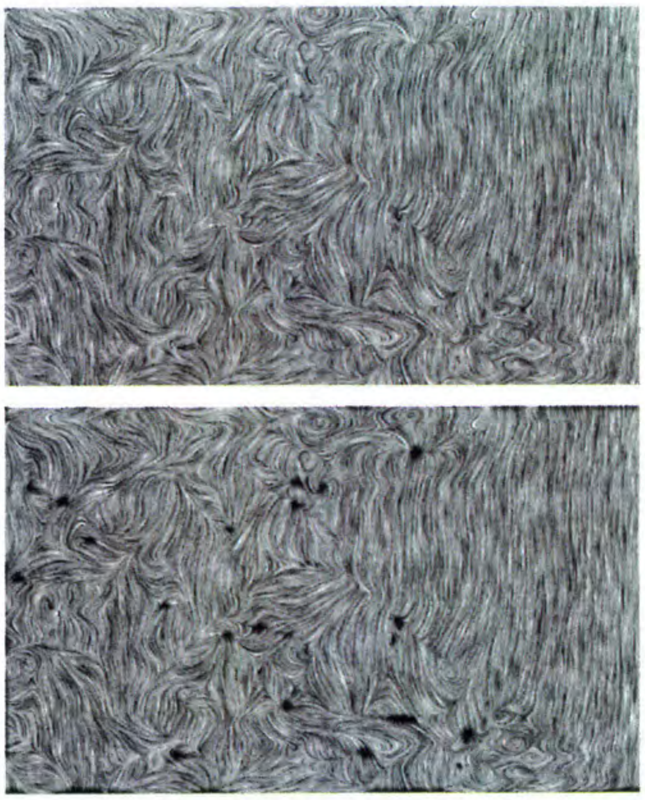


Figure 8: White noise convolved with fluid dynamics vector field using variable normalization (top) versus fixed normalization (bottom).

trast stretching must go to infinity as well. The images in all the figures are contrast stretched.

4.3 IMPLEMENTATION AND APPLICATION

The LIC algorithm is designed as a function which maps an input vector field and texture to a filtered version of the input texture. The dimension of the output texture is that of the vector field. If the input texture is smaller than the vector field the implementation of the algorithm wraps the texture using a toroidal topology. That is, the right and left edges wrap as do the top and bottom edges. If the texture is too large it is cropped to the vector field dimensions. Careful attention must be paid to the size of the input texture relative to that of the vector field. If too small a texture is used, the periodicity induced by the texture tiling will be visible. For scientific applications this is unacceptable. One must insure that the input texture is large enough so that the periodicity induced by the coordinate wrapping is not apparent.

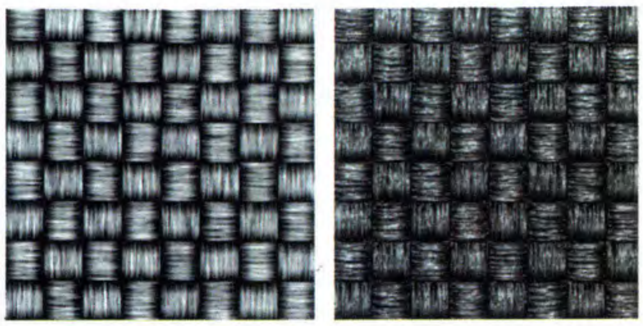


Figure 9: White noise convolved with checkerboard vector field using fixed normalization (left), and then gradient shaded (right) to give the appearance of a rough woven surface texture.

The algorithm can be used as a data operator in conjunction with other operators much like those of Sims[17] and Perlin[14]. Specifically, both the texture and the vector field can be preprocessed and combined with post processing on the output image. The LIC implementation is a module in a data flow system like that found in a number of public domain and commercial products. This implementation allows for rapid exploration of various combinations of operators.

4.3.1 POST PROCESSING

The output of the LIC algorithm can be operated on in a variety of ways. In this section several standard techniques are used in combination with LIC to produce novel results.

An interesting example of constant kernel normalization is shown in figure 9. A simple basket weave pattern is generated by alternating vector directions in a checkerboard fashion. Each checker is surrounded by null vectors. This vector field is then used to convolve white noise. The LIC is truncated as it nears the edges of the checkers which results in a gradual attenuation. When that output is gradient shaded, the basket weave becomes very realistic. While other techniques could be used to generate such a texture, the simplicity of the source data illustrates the versatility of LIC.

A surface wind velocity field is imaged in figure 10 using LIC to blur $1/f$ noise. The resulting image is composed over an image of North America to present scale and location. The LIC algorithm is slightly modified to image vector magnitude by varying the length of the line integral, $2L$, as a function of the vector field magnitude. In figure 10 this effect is seen as clumpiness in $1/f$ cloud-like structures where the wind velocity field is small.

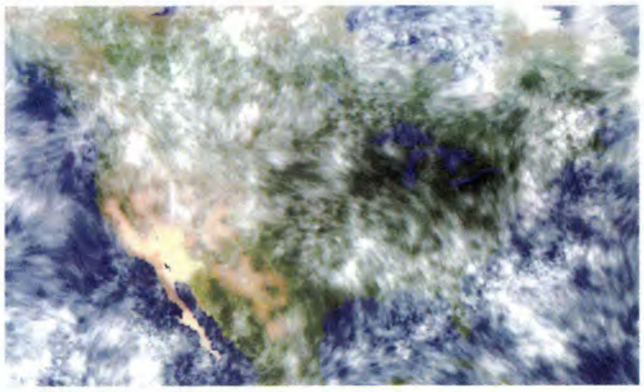


Figure 10: A wind velocity visualization is created by compositing an image of North America under an image of the velocity field rendered using variable length LIC over $1/f$ noise.

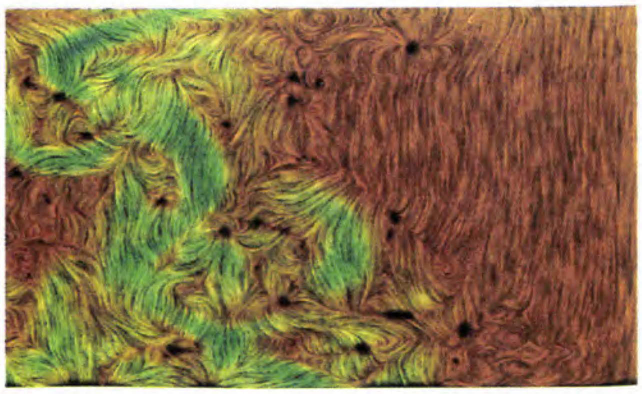


Figure 11: The fixed normalization fluid dynamics field imaged in figure 8 is multiplied by a color image of the magnitude of the vector field.

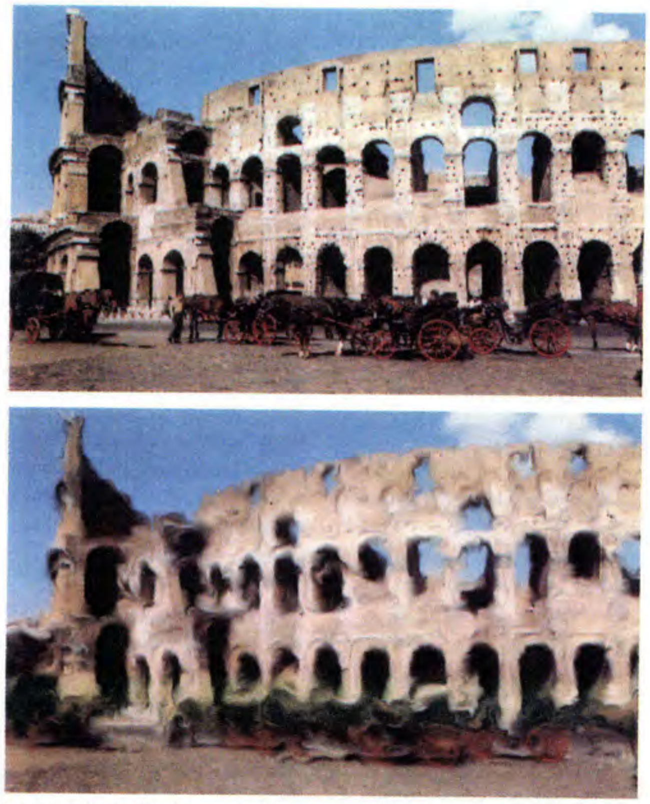


Figure 12: A photograph (top) of the Flavian Amphitheater is used to create a vector field. The field is then used to image a "painted" version of the same photograph (bottom).

Another method to add vector magnitude information is seen in figure 11. The [fixed normalization] fluid dynamics field of figure 8 is multiplied by a color image of the vector magnitude. The advantage of this approach over variable length LIC is that the fine grained detail generated by fixed length LIC is retained even in low magnitude areas.

The LIC algorithm can be used to process an image using a vector field generated from the image itself. In figure 12, a vector field is generated from the input image by low-pass filtering the image, taking the gradient of the resulting image and rotating the vectors by 90°.

The LIC algorithm can also be used to post process images to generate motion blur. A rendering algorithm or paint system can easily specify a pixel by pixel velocity field for objects. By using a biased triangle filter[10] and variable length LIC the input image can be motion blurred in the direction of apparent motion. This has precisely the desired results for motion blurring as seen in figure 13.

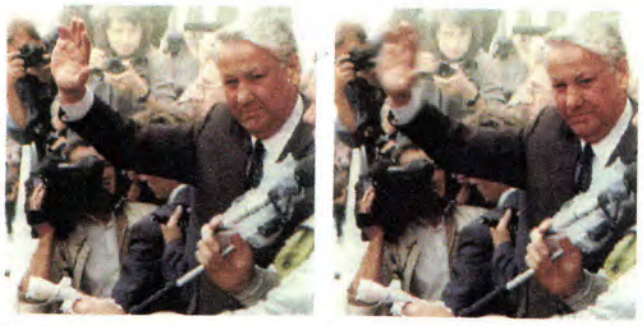


Figure 13: The original photo on the left shows no motion blurring The photo on the right uses variable length LIC to motion blur Boris Yeltsin's waving arm, simulating a slower shutter

4.4 THREE-DIMENSIONAL LIC

The LIC algorithm easily generalizes to higher dimensions. Equations (1), (3) and (5) trivially extend to three dimensions. In the three-dimensional case, cell edges are replaced with cell faces. Both the input vector field and input texture must be three-dimensional. The output of the three-dimensional LIC algorithm is a three-dimensional image or scalar field. This field is rendered using volume rendering techniques such as those found in [21] and [6].

Figure 14 is a three-dimensional rendering of an electrostatic field with two point charges placed a fixed distance apart from one another. In this volumetric rendering, the magnitude of the vector field is used to control the opacity transfer functions. Great efficiency gains can be achieved if the LIC algorithm exploits this by avoiding rendering for vector field cells whose magnitude is outside of the volume renderer's min/max threshold window.

5. PERFORMANCE

There is a distinct performance and quality trade-off between the DDA convolution algorithm and LIC. LIC is roughly an order of magnitude slower than the DDA method. Both algorithms were timed using cells processed per second (CPS) as the figure of merit. The tests were run on an unloaded IBM 550 RISC 6000. The DDA algorithm averages about 30,000 CPS while LIC averages about 3,000 CPS.

The three-dimensional algorithm only marginally degrades in performance with the increase in dimensionality, processing some 1,200 CPS. Since the algorithm remains one-dimensional in nature, the cost per cell only increases by a factor of three as a function of dimension. Using the thresholding described above, the performance of the three-dimensional LIC algorithm has exceeded 30,000 CPS.

6. FUTURE WORK

A number of research directions relating to LIC remain outstanding.

Currently no methods exist for determining the accuracy of a vector field representation, such as those created by LIC or any other method. These accuracy metrics would necessarily be related

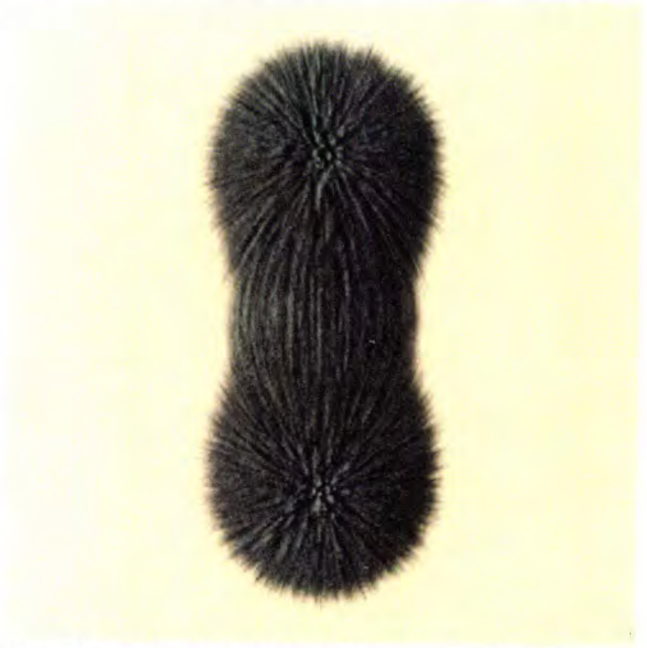


Figure 14: A three-dimensional 512^3 electrostatic field is imaged by volumetrically ray tracing a three-dimensional scalar field produced using LIC over white noise.

to the differential topology of the entire vector field. As mentioned above, much work in theoretical and applied mathematics has been done in this area. This work needs to be studied and applied to efficient vector field imaging algorithms.

LIC is conceptually independent of the advection algorithm used to define the parametric support used by the convolution operation. The method described here might be best characterized as a variable step Euler's method. Other techniques such as a fourth order Runge-Kutta could produce differing or improved results. A thorough investigation into this issue is beyond the scope of this paper. It does, however, represent an area deserving special attention.

Visualizing the orthogonal complement of a two-dimensional vector field is accomplished by rotating the individual vectors 90°. However, in three-dimensional vector fields the orthogonal complement of a vector is a plane. This suggests that a generalization of the one-dimensional LIC filter would be a two-dimensional surface filter. This filter would have as its geometric support a differential surface whose normals would be defined by the vector field, thus creating a Surface Integral Convolution (SIC). As with the LIC, an arbitrary two-dimensional filter could then be used to filter the three-dimensional input image.

Another direction for generalization is to develop versions of the algorithm which operate directly on curvilinear and arbitrarily grided vector fields without resampling the input data. The LIC algorithm could easily be modified to handle arbitrary line intersections and topologies of both type of grids. As with the rectilinear LIC, it would have an analogous three-dimensional generalization. Two additional problems remain however: generating curvilinear and arbitrarily girded textures and output resampling.

One possible image processing application of LIC is the deblurring of motion blurred images. Images acquired with a moving CCD camera often exhibit such blurring. If the CCD frequency response curves and the camera motion are known, one-dimensional deconvolution techniques could be used in conjunction with LIC to deblur the images.

The local nature of the LIC algorithm suggests a parallel implementation. Such an implementation could, in principle, compute all pixels simultaneously. This would allow for interactive generation of periodic motion animations and special effects.

7. SUMMARY

Line integral convolution represents a new and general method for imaging two- and three-dimensional vector fields. The algorithm filters an input image along local stream lines defined by an input vector field and generates an output image. The one-dimensional filter shape is independent of either input and can be arbitrary. To indicate directional flow of the vector field, a whole family of continuous motion filters has been introduced. These filters give apparent motion in the direction of the vector field. The technique can also be used to create special effects. Additionally, the local nature of the algorithm lends itself to efficient and simple implementations.

8. ACKNOWLEDGMENTS

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract W-7405-ENG-48. The SIGGRAPH '93 reviewers provided many helpful comments and suggestions. Thanks to Nelson Max who suggested using higher order functions within a cell and who provided critical assessment all along the way. Roger Crawfis deserves special thanks for various conversations over the past couple of years on the topic of vector visualization. Chuck Grant provided helpful suggestions clarifying the language used to discuss periodic motion filters. John Bell and Jeff Greenough provided the turbulent computational fluid dynamics data used in figures 2, 4, 8 and 11 and for using the algorithm in their work. Dean Williams and Jerry Potter provided the North America wind velocity data. Lastly, thanks to John Zych who helped with the rendering of the North America image.

REFERENCES

1. Bresenham, J. Algorithm for Computer Control of a Digital Plotter. In *IBM Systems Journal* 4, 1 (1965), 25-30.
2. Bronstein, I. and Semendyayev, K. *Handbook of Mathematics*. Van Norstrand Reinholt (1985), 291-293.
3. Chang, S. *Fundamentals Handbook of Electrical Engineering and Computer Engineering*. John Wiley & Sons, Inc. (1982), 264-266.
4. Chui, K. *An Introduction to Wavelets*. Academic Press, Inc. (1992), 49-60.
5. Crawfis, R. and Max, M. Direct Volume Visualization of Three-Dimensional Vector Fields. *Proceedings of the Workshop on Volume Visualization*, Kaufman and Lorensen Eds (1992).
6. Drebin, R., Carpenter, L. and Hanaran, P. Volume Rendering. *Computer Graphics* 22, 4 (August 1988), 65-74.
7. Dumortier, F., Roussarie, R., Sotomayor, J. and Zoladek, H., Study of Field Bifurcations. *Lecture Notes in Mathematics*, Springer-Verlag (1991).
8. Freeman, W., Adelson, E. and Heeger, D. Motion without Movement. *Computer Graphics* 25, 4 (July 1991), 27-30.
9. Haeberli, P. Paint By Numbers: Abstract Image Representation. *Computer Graphics* 24, 4 (August 1990), 207-214.
10. Heckbert, P. Filtering by Repeated Integration. *Computer Graphics* 20, 4 (August 1986), 315-321.
11. Kajiya, J. and Kay, T. Rendering Fur with Three Dimensional Textures. *Computer Graphics* 23, 3 (July 1989), 271-280.
12. Kenwright, D. and Mallinson, G. A 3-D Streamline Tracking Algorithm Using Dual Stream Functions. *IEEE Visualization '92 Conference Proceedings* (October 1992), 62-68.
13. Max, Nelson. Personal Communication (1992).
14. Perlin, K. An Image Synthesizer. *Computer Graphics* 19, 3 (August 1985), 287-296.
15. Perlin, K. Hypertexture. *Computer Graphics* 23, 3 (July 1989), 253-262.
16. Pratt, W. *Digital Image Processing*. 2nd ed. John Wiley & Sons, Inc. (1991), 243-245.
17. Sims, K. Artificial Evolution for Computer Graphics. *Computer Graphics* 25, 4 (August 1991), 319-328.
18. Sims, K. Choreographed Image Flow. *The Journal of Visualization and Computer Animation* 3, 1 (January-March 1992), 31-43.
19. Tufte, E. The Visual Display of Quantitative Information. *Chesire, CT: Graphics Press* (1983).
20. Turk, G. Generating Textures on Arbitrary Surfaces Using Reaction-Diffusion Textures. *Computer Graphics* 25, 4 (July 1991), 289-298.
21. Upson, C. and Keeler, M. V-Buffer: Visible Volume Rendering. *Computer Graphics* 22, 4 (August 1988), 59-64.
22. Van Gelder, A. and Wilhelms, J. Interactive Animated Visualization of Flow Fields. *Proceedings of the Workshop on Volume Visualization*, Kaufman and Lorensen Eds. (1992).
23. Van Wijk, J. Spot Noise Texture Synthesis for Data Visualization. *Computer Graphics* 25, 4 (July 1991), 309-318.
24. Witkin, A. and Kass, M. Reaction-Diffusion Textures. *Computer Graphics* 25, 4 (July 1991), 299-308.

Frequency Domain Volume Rendering

*Takashi Totsuka** *Marc Levoy*†

*SONY Corporation
†Computer Science Department, Stanford University

Abstract

The Fourier projection-slice theorem allows projections of volume data to be generated in $O(n^2 \log n)$ time for a volume of size n^3. The method operates by extracting and inverse Fourier transforming 2D slices from a 3D frequency domain representation of the volume. Unfortunately, these projections do not exhibit the occlusion that is characteristic of conventional volume renderings. We present a new frequency domain volume rendering algorithm that replaces much of the missing depth and shape cues by performing shading calculations in the frequency domain during slice extraction. In particular, we demonstrate frequency domain methods for computing linear or nonlinear depth cueing and directional diffuse reflection. The resulting images can be generated an order of magnitude faster than volume renderings and may be more useful for many applications.

CR Categories: I.3.7 [Computer Graphics]: Three-dimensional Graphics and Realism.; I.3.3 [Computer Graphics]: Picture/Image Generation; Display Algorithms.

Additional Keywords: Volume rendering, Fourier transform, Shading models, Scientific visualization, Medical imaging, Digital signal processing.

1 Introduction

Volume rendering is an important tool for visualizing 3D scalar fields. Most existing algorithms operate in the spatial domain. They can be classified as either image space algorithms (e.g. [7]) or object space algorithms (e.g. [4], [15]) depending on the order in which the data is traversed: along each ray cast from the image plane or along X, Y, and Z axis of the volume data. The complexity of these algorithms is $O(n^3)$ since all voxels must be visited to render an image. This high cost limits the use of these algorithms in interactive environments. Although efficient algorithms exist for sparse data sets [8], [14],[16], such optimization is data dependent.

In an effort to drastically reduce rendering costs, frequency domain algorithms based on the Fourier projection slice theorem have been proposed [5], [10]. It is well known that the integral of a 1D signal is equal to the value of its spectrum at the origin. The Fourier projection slice theorem extends this notion to higher dimensions. For a 3D volume, the theorem states that the following two are a Fourier transform pair:

- The 2D image obtained by taking line integrals of the volume along rays perpendicular to the image plane.
- The 2D spectrum obtained by extracting a slice from the Fourier transform of the volume along a plane which includes the origin and is parallel to the image plane.

Using this theorem, once a volume data is Fourier transformed, an (orthographic) image for any viewing direction can be obtained by extracting a 2D slice of the 3D spectrum at the appropriate orientation and then inverse Fourier transforming it (figure 1). The cost of this approach is dominated by the 2D inverse fast Fourier transform (IFFT) which is $O(n^2 \log n)$. Hence, the overall cost is also $O(n^2 \log n)$. Since $\log n$ grows slowly, the advantage of this approach over spatial domain algorithms is greater at large data sizes.

Despite their theoretical speed advantage, frequency domain volume rendering algorithms suffer from several well-known problems:

High interpolation cost: Because the sample points of the 3D spectrum and those of the 2D slice do not coincide except at the origin, the 3D spectrum must be interpolated and then resampled in order to extract a 2D slice. Since this interpolation is imperfect, replicas of the volume data are not fully suppressed, causing ghosts to appear on the projection image. Because any

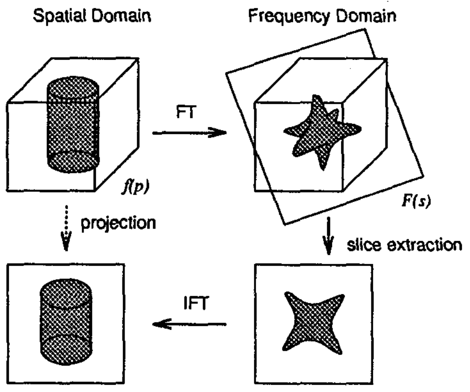


Figure 1: Volume rendering using Fourier projection slice theorem

* Sony Corporation. 6-7-35 Kitashinagawa, Shinagawa Tokyo 141, Japan (totsuka@av.crl.sony.co.jp)
† Center for Integrated Systems, Stanford University Stanford, CA 94305 (levoy@cs.stanford.edu)

filter that provides a sharp cutoff in the spatial domain also has wide support, high-quality interpolation is expensive. As the interpolation is $O(n^2)$, the FFT is still asymptotically dominant. However, due to a large constant factor associated with the interpolation, current implementations spend the majority of their running time in interpolation, making the algorithm not attractive for practical data sizes (128^3 or 256^3).

Memory cost: Due to the wide dynamic range and complex arithmetic associated with Fourier transforms, a pair of floating point numbers is required for each voxel. Assuming a 64-bit double precision representation, 16 bytes are required per voxel. By contrast, only 1 byte per voxel is necessary in spatial domain algorithms.

Lack of depth information: The projection obtained by the Fourier projection slice theorem is a line integral normal to the direction of view. Voxels on a viewing ray contribute equally to the image regardless of their distance from the eye. The image therefore lacks occlusion, an important visual cue. While some users (diagnostic radiologists in particular) prefer integral projections since nothing is hidden from view, this characteristic would be considered a drawback in most applications.

The first two problems listed above are technical in nature, and several promising solutions are proposed later in this paper. The lack of occlusion is fundamental, however, in so far as no projection-slice theorem is known that mimics the integro-differential equation ([6]) approximated by volume rendering algorithms. Fortunately, occlusion is only one of many cues employed by the human visual system to determine the shape and spatial relationships of objects. Other available cues include perspective, shading, texture, shadows, atmospheric attenuation, stereopsis, ocular accommodation, head motion parallax, and the kinetic depth effect.

It is possible, of course, to apply any shading technique in the spatial domain before the volume is Fourier transformed. However, such a naive approach would require recomputation of the volume followed by an expensive 3D forward FFT each time the view or the lighting condition is changed.

In an earlier paper [9], we instead showed that for a limited class of shading models, the dependence on viewing direction and lighting direction could be factored out of the projection integral, yielding equations of the form

$$I = \sum_{i=0}^{n} w_i \left(\int_{-\infty}^{+\infty} f_i\big(x(t), y(t), z(t)\big)\, dt \right). \qquad (1)$$

Here, effects of viewing and lighting direction are solely expressed by weights w_i while the volumes f_i are independent of them. The indicated integration can be evaluated efficiently using the projection slice theorem. For example, linear depth cueing can be computed as the weighted sum of projections through three volumes that are depth cued before 3D forward FFT along X, Y, and Z directions, respectively.

The obvious disadvantage of this hybrid spatial-frequency domain approach is that it requires multiple copies of the volume. While still asymptotically faster than conventional spatial domain volume rendering, implementation considerations (problems one and two above) make it barely superior in practice.

In the present paper, we describe methods for rendering volumes with depth cueing and directional shading that operate entirely within the frequency domain. They are based on two well-known properties of the Fourier transform.

- Multiplication by a linear ramp in the spatial domain is equivalent to differentiation in the Fourier domain.
- Differentiation in the spatial domain is equivalent to multiplication by a linear ramp in the Fourier domain.

Using these properties, depth cueing implemented in [9] as spatial domain multiplication, is implemented in the present paper using frequency domain differentiation. Similarly, directional shading, implemented in [9] using spatial domain differentiation, is implemented in the present paper using frequency domain multiplication.

The remainder of the paper is organized as follows. Section 2 reviews the previous works. Section 3 presents our new frequency domain shape cueing techniques. Sections 4 and 5 refer to solutions to the interpolation and the memory cost problems, respectively. Section 6 shows results from our implementation, and section 7 gives conclusions and possible future directions.

2 Base Algorithm

We begin by briefly reviewing current frequency domain volume rendering algorithms. In the following discussion, small letters ($f, g, \ldots$) represent data in the spatial domain and capital letters ($F, G, \ldots$) represent data in the frequency domain. We also assume that the transform between the two domains is the Fourier transform which is denoted by $\mathcal{F}$.

Let $f(\boldsymbol{x})$ be a volume and $F(\boldsymbol{s})$ be its Fourier transform. $\boldsymbol{x}$ and $\boldsymbol{s}$ are 3D vectors in the spatial and frequency domain, respectively. Given $f(\boldsymbol{x})$, the algorithm first transforms it into the frequency domain to yield $F(\boldsymbol{s})$. This is done only once. For each view, the discrete spectrum $F(\boldsymbol{s})$ is interpolated along the extraction plane (parallel to the image plane and passing through the origin) using a filter $H(\boldsymbol{s})$. The interpolated spectrum is resampled to obtain a 2D spectrum which is then inverse transformed to obtain a spatial domain projection.

By the convolution theorem, interpolation $F(\boldsymbol{s}) * H(\boldsymbol{s})$ corresponds to $f(\boldsymbol{x}) \cdot h(\boldsymbol{x})$ in the spatial domain. Here, $h(\boldsymbol{x})$ is the response of the filter. Unless $H(\boldsymbol{s})$ is an ideal lowpass filter, its response has a smooth shoulder. Thus, the periphery of the volume and consequently the periphery of the projected image is attenuated. To cope with this "vignetting" problem, the volume data $f(\boldsymbol{x})$ can be premultiplied by the reciprocal of the response, $p_m(\boldsymbol{x}) = \frac{1}{h(\boldsymbol{x})}$ before its forward transformation [10]. As H and P_m cancel during interpolation, we obtain a correct slice of F (figure 2). We have implemented this method using filters obtained from Malzbender and have obtained excellent results, as documented in section 4 and 6.

3 Shape Cueing Techniques

3.1 Depth Cueing

Depth cueing is obtained by weighting voxels according to their distance from the observer. Let $d(\boldsymbol{x})$ be the weighting function or depth cueing function for a given eye position. Then, a depth-cued volume is expressed as $f(\boldsymbol{x}) \cdot d(\boldsymbol{x})$. By transforming it to the frequency domain and extracting a slice, we obtain a depth cued projection. As stated earlier, this straightforward approach requires an expensive 3D FFT ($n^3 \log n$) for each view. There is, however, an elegant and inexpensive equivalent operation in frequency domain.

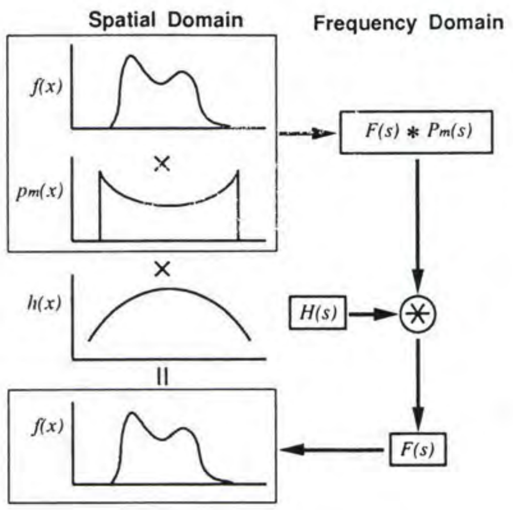


Figure 2: Premultiplication of the volume data

Including the compensation $p_m(\boldsymbol{x})$ for the filter response, spatial domain depth cueing can be expressed as $f(\boldsymbol{x}) \cdot d(\boldsymbol{x}) \cdot p_m(\boldsymbol{x})$. By transforming and interpolating, this corresponds to $\mathcal{F}\{f(\boldsymbol{x}) \cdot d(\boldsymbol{x}) \cdot p_m(\boldsymbol{x})\} * H(\boldsymbol{s})$ at sample points on the slice in the frequency domain. Using the convolution theorem, this expression can be rewritten as follows:

$$\begin{aligned} & \mathcal{F}\{f(\boldsymbol{x})\, d(\boldsymbol{x})\, p_m(\boldsymbol{x})\} \; * \; H(\boldsymbol{s}) \\ = \; & (F(\boldsymbol{s}) * D(\boldsymbol{s}) * P_m(\boldsymbol{s})) \; * \; H(\boldsymbol{s}) \\ = \; & (F(\boldsymbol{s}) * P_m(\boldsymbol{s})) \; * \; (H(\boldsymbol{s}) * D(\boldsymbol{s})) \\ = \; & \mathcal{F}\{f(\boldsymbol{x})\, p_m(\boldsymbol{x})\} \; * \; H'(\boldsymbol{s}) \qquad (2) \end{aligned}$$

where $H'(\boldsymbol{s}) = H(\boldsymbol{s}) * D(\boldsymbol{s})$.

Thus, merely by replacing the interpolation filter H with H', we have obtained depth cueing. Note that the above expression operates entirely in the frequency domain, and moreover is evaluated only on the plane of the slice being extracted. Hence, it is a 2D operation. Note also that because $\mathcal{F}\{f(\boldsymbol{x}) \cdot p_m(\boldsymbol{x})\}$ is independent of the eye position, the 3D forward transform is performed only once.

Although H' must be computed for each view, the cost of recomputation is small because the support of filter H is small ($3^3 \sim 5^3$) and $D(\boldsymbol{s})$ is usually a simple expression. In practice, the recomputation is negligible compared with the cost of interpolation itself.

This frequency domain depth cueing method applies to any depth cueing function $d(\boldsymbol{x})$. Indeed, the method can be designed to highlight the middle portion of the volume while attenuating the front and back portions.

By way of example, we first consider simple linear depth cueing, $d_l(\boldsymbol{x})$. Let the view vector be $\boldsymbol{V}$. The signed depth measured from the origin of the volume is thus given by $(\boldsymbol{V} \cdot \boldsymbol{x})$, and $d_l(\boldsymbol{x})$ can be written as

$$d_l(\boldsymbol{x}) = C_{cue}(\boldsymbol{V} \cdot \boldsymbol{x}) + C_{avg} \qquad (3)$$

where C_{cue} is the strength of the depth cueing effect and C_{avg} is a constant (see figure 3). Taking Fourier transforms, we obtain

$$D_l(\boldsymbol{s}) = -\frac{C_{cue}}{i2\pi}(\boldsymbol{V} \cdot \boldsymbol{\Delta}) + C_{avg}\,\delta(\boldsymbol{s}) \qquad (4)$$

where $\boldsymbol{\Delta} = [\Delta_x, \Delta_y, \Delta_z]$ is the differential operator of convolution ($\Delta_x * f = \frac{\partial}{\partial x} f$). Substituting the interpolation filter

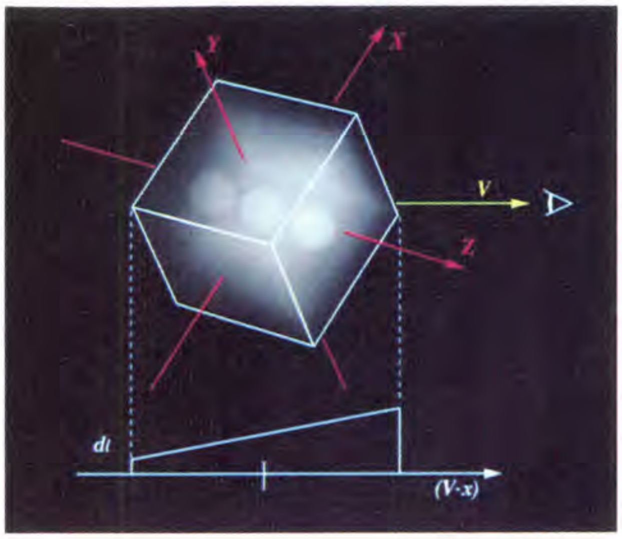


Figure 3: Linear depth cueing

with depth cueing (H') yields

$$\begin{aligned} H'(\boldsymbol{s}) \; &= \; H(\boldsymbol{s}) * D_l(\boldsymbol{s}) \\ &= \; -\frac{C_{cue}}{i2\pi}(\boldsymbol{V} \cdot \nabla H(\boldsymbol{s})) + C_{avg}\, H(\boldsymbol{s}) \qquad (5) \end{aligned}$$

The first term exhibits the depth cueing effect. Since ∇H can be precomputed and stored in a table, computation of H' is of insignificant cost. An example of frequency domain linear depth cueing and projection is shown in figure 6(b). As a reference, the same volume rendered without depth cueing is shown in figure 6(a).

Although any function can be used for D, finding one that has a simple form reduces the cost of computing H'. The size of H' is also a consideration, since it directly impacts rendering time. To illustrate this important issue, let us employ a half period of a sine wave as $d(\boldsymbol{x})$. Since the transform of a sine function is two impulses, H' can be computed by shifting H and adding three copies[1] with complex weights. Note that this considerably increases the size of the filter kernel. By adjusting the origin, amplitude, and period such that the value is zero at the farthest voxel and unity at the closest voxel, we eliminate the need for a DC term. D now has the form $C_1\,\delta(\boldsymbol{s} - \boldsymbol{s}_w) + C_2\,\delta(\boldsymbol{s} + \boldsymbol{s}_w)$ where C_1 and C_2 are complex constants determined by the amplitude and the shift of the wave and $\boldsymbol{s}_w$ is determined by the period of the wave. The period is typically made long enough so that the depth cueing appears almost linear. We can further remove one of the impulses by doubling the weight of the remaining impulse. By removing one of the impulses, the projection image is no longer a real[2]. However, the real part of the result still contains the correct projection image. With this technique, depth cueing is implemented by an interpolation with a shifted H, which is practically free.

The notion of a shifted H gives us an alternative way to look at the process. Extracting a slice from a spectrum at a position translated from the origin by a distance d in a

[1]Two for the impulses of the sine wave term and one for the constant term of $d(\boldsymbol{x})$.

[2]The imaginary part is a cosine wave since we are using the analytic signal of the depth cueing function. See the discussion on the Hilbert transform in [1].

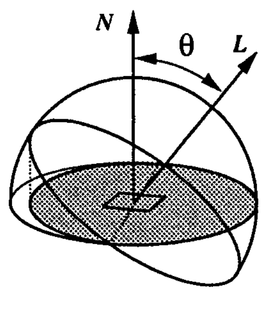


Figure 4: Hemispherical light source

direction $\boldsymbol{V}$ corresponds to phase-shifting the spatial domain projection by $e^{i2\pi d t}$ at distance t in the same direction $\boldsymbol{V}$. The real part of such a phase-shifted projection appears to fade in and out as a function of position in direction $\boldsymbol{V}$ and, for appropriate values of d, the visual effect is that of depth cueing.

3.2 Directional Shading

In a scene composed of surfaces, directional shading using the well-known Lambertian reflection model is given by

$$C_{amb}\, O_c L_{amb} + C_{dif}\, O_c L_{dif}\, \mathrm{MAX}\,(0, (\boldsymbol{N} \cdot \boldsymbol{L})) \qquad (6)$$

where C_{amb} and C_{dif} are constants defining the strength of ambient and directional shading terms, O_c is an object color, L_{amb} and L_{dif} are constants defining the color of ambient and directional lights, and $\boldsymbol{N}$ and $\boldsymbol{L}$ are unit surface normal and light vectors, respectively.

Ignoring the attenuation of light inside the volume, the ambient term can be approximated using

$$C_{amb}\, L_{amb}\, f(\boldsymbol{x}) \qquad (7)$$

The diffuse term, however, must be handled carefully because the nonlinear function MAX does not have a simple frequency domain representation. Note that the frequently used alternative, $|\boldsymbol{N} \cdot \boldsymbol{L}|$, which shades surfaces as if they are two-sided rather than the bounding surface of a solid, is also nonlinear and cannot be handled directly in the frequency domain.

To avoid this problem, we employ a hemispherical light source [12], [9]. The irradiance E_i on a surface having normal vector $\boldsymbol{N}$ illuminated by a hemisphere whose pole points in direction $\boldsymbol{L}$ as shown in figure 4 is proportional by Nusselt's analog (as described in [3]) to the projection of the visible portion of the hemisphere down onto the plane containing the surface, or

$$E_i = L_{dif}\,\frac{1}{2}(1 + \cos\theta) = L_{dif}\,\frac{1}{2}(1 + (\boldsymbol{N} \cdot \boldsymbol{L})) \qquad (8)$$

With this shading model, the diffuse term in a surface model is expressed as

$$C_{dif}\, O_c L_{dif}\,\frac{1}{2}(1 + (\boldsymbol{N} \cdot \boldsymbol{L})) \qquad (9)$$

For volumes, we have

$$\begin{aligned} & C_{dif} L_{dif}\,\frac{1}{2}\,|\nabla f(\boldsymbol{x})| \left(1 + \frac{(\nabla f(\boldsymbol{x}) \cdot \boldsymbol{L})}{|\nabla f(\boldsymbol{x})|}\right) \\ = \; & C_{dif} L_{dif}\,\frac{1}{2}\,(|\nabla f(\boldsymbol{x})| + (\nabla f(\boldsymbol{x}) \cdot \boldsymbol{L})) \qquad (10) \end{aligned}$$

Since volume datasets do not have explicitly defined surfaces, $\nabla f(\boldsymbol{x})$ is used as the normal vector at each location. The strength of directional shading in volume rendering algorithms is commonly made proportional to the gradient magnitude as a simulation of the *surface-ness* of the volume [4],[7]. Locales having high gradient magnitudes (i.e., steep jumps in density) reflect more light.

Equation (10) can be computed entirely in the frequency domain. By the derivative theorem, the gradient in one domain is the first moment in the other domain. Thus, the shading computation can be performed as a moment computation in the frequency domain. This useful property of linear shading can also be exploited in image understanding algorithms. For example, [13] uses the moment to estimate the orientation of surfaces assuming that the reflectance function is linear with respect to the slope of the surfaces.

Transforming equations (7) and (10) to the frequency domain and including compensation for the filter response, we obtain

$$\begin{aligned} & \mathcal{F}\{C_{amb}\, L_{amb}\, f(\boldsymbol{x}) \\ & \qquad + C_{dif} L_{dif}\,\frac{1}{2}\,(|\nabla f(\boldsymbol{x})| + (\nabla f(\boldsymbol{x}) \cdot \boldsymbol{L}))\} \\ = \; & (C_{amb}\, L_{amb} + i\pi C_{dif}\, L_{dif}\,(\boldsymbol{s} \cdot \boldsymbol{L})) \\ & \qquad \times (\mathcal{F}\{f(\boldsymbol{x})\, p_m(\boldsymbol{x})\} * H(\boldsymbol{s})) \\ & + \frac{1}{2} C_{dif}\, L_{dif}\,(\mathcal{F}\{|\nabla f(\boldsymbol{x})|\, p_m(\boldsymbol{x})\} * H(\boldsymbol{s})) \quad (11) \end{aligned}$$

The first term corresponds to the ambient term and the $(\boldsymbol{N} \cdot \boldsymbol{L})$ part of equation (9) while the second term corresponds to the accompanying constant 1. Once $f(\boldsymbol{x})\, p_m(\boldsymbol{x})$ and $|\nabla f(\boldsymbol{x})|\; p_m(\boldsymbol{x})$ are Fourier transformed, the shading computation can be performed during slice extraction (figure 5). Note that the interpolation filter H is applied first in order to reconstruct the pure spectrum of $f(\boldsymbol{x})$ from the premultiplied volume. Then, the first moment of the spectrum is computed to apply the directional shading.

Although computing a moment incurs a few additional floating point operations per sample on the slice, the additional expense is small relative to the number of operations that are required to evaluate the convolution at the sample point. It should also be noted that equation (11) can be easily extended to multiple light sources. In this case, we only have to add the moment terms for additional light sources. The increase in the computation cost is minor.

Figure 6(c) shows a projection shaded using this technique. As before, the method operates entirely in the frequency domain and requires computations only on the plane of the slice being extracted

The major drawback of this shading model is that it requires a second spectrum, $\mathcal{F}\{|\nabla f(\boldsymbol{x})|\; p_m(\boldsymbol{x})\}$ since there is no simple way to compute a gradient magnitude in the frequency domain. Hence, two slices must be extracted from two volumes. A linear shading equation such as $C_{amb}\, L_{amb}\, f(\boldsymbol{x}) + C_{dif} L_{dif}\, \nabla f(\boldsymbol{x})$ that requires only one volume can be derived under an appropriate interpretation. However, the upper bound of C_{dif} is restricted in order not to generate negative values and consequently the shading effect is restricted.

3.3 Combining Depth Cueing and Shading

It is possible to combine the depth cueing and directional shading techniques described in the foregoing section. When the two techniques are used together, the shading must be

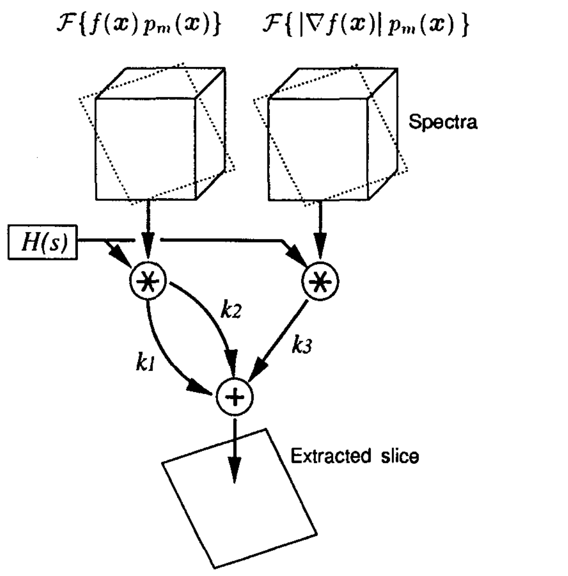

$\mathcal{F}\{f(x)\,p_m(x)\}$
$\mathcal{F}\{\,|\nabla f(x)|\,p_m(x)\,\}$
Spectra
H(s)
k2
k3
k1
Extracted slice

Figure 5: Shading computation in frequency domain. $k_1 = C_{amb}\,L_{amb}$ (ambient term), $k_2 = i\pi C_{dif}\,L_{dif}\,(s \cdot \boldsymbol{L})$ (shading term), $k_3 = \frac{1}{2}C_{dif}\,L_{dif}$ (constant term).

applied first. Otherwise, distortion by the depth cueing would result in incorrect gradient vector by which the shading effect is computed. However, this order of operation requires two convolutions: one performed before the shading computation to recover F by interpolation filter H and one performed after shading in order to apply the depth cueing function. This approach makes depth cueing no longer an inexpensive operation since we can't use the composite filter H'.

We can work around this problem by reversing the order of shading and depth cueing and then adjusting the result to get the desired effect. Using this ordering, we employ the composite filter H' to perform the interpolation and the depth cueing at once. As we will see, for practical settings, even this adjustment is not necessary.

Here, we will examine the effect of reversed order operation in spatial domain. We focus on the gradient term of the shading equation (second term of equation (10)) since other terms are not affected by the order. Applying depth cueing function $d(\boldsymbol{x})$ to equation (10), we obtain the shaded and depth cued term. Omitting the coefficient $\frac{1}{2}C_{dif}\,L_{dif}$, the gradient term is $(\nabla f(\boldsymbol{x}) \cdot \boldsymbol{L}\,)\,d(\boldsymbol{x})$. Reversing the order of computation, we get

$$\begin{aligned} & (\nabla[f(\boldsymbol{x})\,d(\boldsymbol{x})] \cdot \boldsymbol{L}\,) \\ = \; & (\nabla f(\boldsymbol{x}) \cdot \boldsymbol{L}\,)\,d(\boldsymbol{x}) + f(\boldsymbol{x})(\nabla d(\boldsymbol{x}) \cdot \boldsymbol{L}\,) \end{aligned} \tag{12}$$

The second term is the difference from the correct value. Since $d(\boldsymbol{x})$ is a function of depth $(\boldsymbol{V} \cdot \boldsymbol{x})$, the difference can be rewritten as

$$\begin{aligned} & f(\boldsymbol{x})\,(\nabla[d_{1D}\,(\boldsymbol{V} \cdot \boldsymbol{x})] \cdot \boldsymbol{L}\,) \\ = \; & f(\boldsymbol{x})\,d'_{1D}\,(\boldsymbol{V} \cdot \boldsymbol{x})\;(\boldsymbol{V} \cdot \boldsymbol{L}\,) \end{aligned} \tag{13}$$

where $d_{1D}(t)$ is a 1D depth cueing function. To maximize the shading effect, $\boldsymbol{L}$ is usually set perpendicular to $\boldsymbol{V}$ (i.e., the scene is illuminated from the side). In this case, the difference term becomes zero and the adjustment is not necessary. An example of this common special case is shown in figure 6(d).

If $(\boldsymbol{V} \cdot \boldsymbol{x})$ is non-zero, we need an adjustment. For linear depth cueing, the difference term including all the coefficients is

$$\frac{1}{2}C_{cue}C_{dif}\,L_{dif}\,f(\boldsymbol{x})\,(\boldsymbol{V} \cdot \boldsymbol{L}) \tag{14}$$

which we can compute during slice extraction without convolution. For a more complex depth cueing function, a convolution is necessary.

4 Reducing Rendering Time

Although the interpolation required in order to extract an arbitrarily oriented slice from the 3D spectrum is $O(n^2)$, it consumes most of the running time. As might be expected, the cost of this interpolation step is almost entirely determined by the size of the filter. For the $3 \times 3 \times 3$ filter we employ, 27 input samples contribute to each output sample. If we instead employed a $1 \times 1 \times 1$ filter, only one input sample would contribute to each output sample, a great saving in time. Because a smaller filter has less sharp cut off in spatial domain, the resulting image would contain strong ghosts if it were used uniformly over the entire interpolation process. However, by adaptively changing the filter size, we can reduce rendering time while maintaining high image quality.

Most of the energy in a spectrum usually resides in a small number of low frequency components, while the vast majority of high frequency components are nearly zero. We have observed that usually 99% of the energy is contained by about 10% of the frequency components.

This property makes an adaptive scheme which selects an inexpensive filter for weak frequency components very attractive. For simplicity, let us consider interpolation of a 1D spectrum F by two filters; a larger filter H_1 and a smaller filter H_2. Each input sample component is filtered or scattered by either H_1 or H_2 according to its strength. Let F_1 be the set of those samples that are filtered by H_1 and F_2 be those filtered by H_2. Obviously, $F_1 + F_2 = F$. The correct result we want is $F * H_1$ or in the spatial domain, $f\,h_1$. The adaptive scheme can thus be written as follows:

$$\begin{aligned} & \mathcal{F}^{-1}\{\,F_1 * H_1 \;+\; F_2 * H_2\,\} \\ = \; & \mathcal{F}^{-1}\{\,F * H_1 \;+\; F_2 * (H_2 - H_1)\,\} \\ = \; & f\,h_1 + f_2\,(h_2 - h_1) \end{aligned} \tag{15}$$

The term $f_2\,(h_2 - h_1)$ denotes the difference between the adaptively filtered image and the correct image. The mean square error is given by integrating the power of this error term. Using Rayleigh's theorem, its upper bound is given in the frequency domain as follows.

$$\begin{aligned} & \frac{1}{L}\int_{-\infty}^{+\infty} |\,f_2\,(h_2 - h_1)\,|^2\,dx \\ \leq \; & \frac{1}{L}\,h_{d\text{-}max}^2 \int_{-\infty}^{+\infty} |\,f_2\,|^2\,dx \\ = \; & \frac{1}{L}\,h_{d\text{-}max}^2 \int_{-\infty}^{+\infty} |\,F_2\,|^2\,ds \end{aligned} \tag{16}$$

where L is the length of the non-zero region of f and $h_{d\text{-}max}$ is the maximum of $|h_2 - h_1|$. This upper bound allows us to select input samples to be filtered by H_2 such that the mean square error of the rendered image is below a user defined tolerance. Similar analysis provides an upper bound for the mean square error when more than 2 filters are employed. The idea extends straightforwardly to 3D discrete signals.

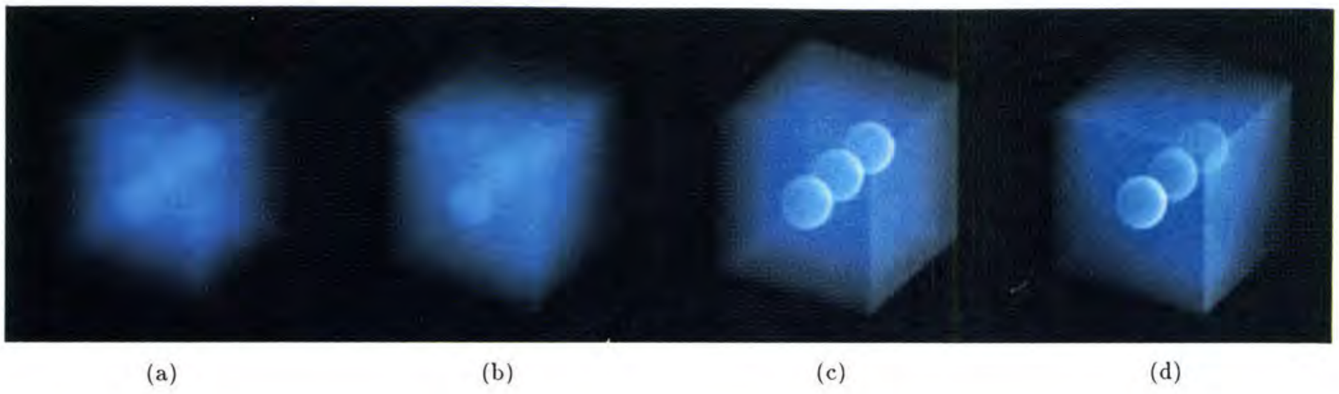


Figure 6: Examples of frequency domain depth cueing and shading. (a) projection without depth cueing, (b) linear depth cueing, (c) directional shading without depth cueing, (d) directional shading with depth cueing.

This adaptive scheme is incorporated to the slice extraction as follows. First, each sample in the 3D spectrum is examined, and those whose magnitude is small enough to satisfy equation (16) are marked. This process is done only once after a volume data is transformed to the frequency domain. During slice extraction, each sample point on the slice plane is visited. If for a given sample point all of the 3D spectrum voxels that fall within the support of the larger filter are marked, the smaller filter is employed instead.

It is possible to improve this scheme further. To avoid testing all voxels falling within the support of the larger filter, we modify the preprocess to mark only those voxels that themselves satisfy equation (16) and for which all neighboring voxels lying within a distance from them equal to one-half of the support of the larger filter satisfy the equation. Given this more conservative marking, it is sufficient during slice extraction to test the spectrum voxel closest to the slice sample position. If that voxel is marked, we know without visiting any other voxels that it is safe to employ the smaller filter.

5 Reducing Memory Cost

Because the 3D spectrum is complex and requires a floating point representation due to its large dynamic range, a straightforward implementation using a double precision format consumes 16 times more memory than a spatial domain algorithm[3]. This explosion in memory cost can be controlled by using the Hartley transform [10] and a shorter number representation.

The Hartley transform is a direct relative of the Fourier transform [2]. The transform is defined as follows:

$$\mathcal{H}\{f(x)\} = F_H(s) = \int_{-\infty}^{+\infty} f(x)\,\mathrm{cas}2\pi sx\,dx \qquad (17)$$

where $\mathrm{cas}2\pi sx = \cos 2\pi sx + \sin 2\pi sx$. Since the kernel is a real function, this transform maps a real function $f(x)$ to a real spectrum $F_H(s)$. Use of the Hartley transform, therefore, eliminates the need for a complex number. Since the Fourier spectrum of a real signal is hermitian[4], the same amount of memory saving is possible with the Fourier transform by dropping half of the spectrum (e.g., store only the positive coefficients along the S_x axis). However, such implementation would unnecessarily complicate the slice extraction process.

Due to wide dynamic range of spectra, a floating point format is necessary. Considering the necessity of premultiplying the volume before transforming, a 64-bit double precision format is a safe choice to represent a spectrum of a 256^3 volume. However, even using the Hartley transform, this occupies 8 times more memory than the original volume. This problem can be minimized by using a shorter floating point format. We have defined and used a 16-bit floating point format which reduces the memory cost factor to two.

6 Results

Figures 7-9 show images rendered using the algorithms we have described. The shading, depth cueing, adaptive filtering, the Hartley transform, and the 16-bit floating point format are all used in rendering these three images.

Figure 7 shows a human skull mounted in a lucite head cast. The data was acquired using computed tomography (CT). Zeros are padded to the original data (106^3) and resulting 128^3 volume data was rendered. The volume is shaded by a hemispherical light source located to the right and is also linearly depth cued with respect to the observer's position.

The use of multiple light sources is shown in figure 8. A polygonalization of the Utah teapot has been 3D scan-converted into a 256^3 volume data which is then shaded by a red, a green, and a blue light located perpendicular to the observer and 120 degrees apart. The resulting color on the surface provides some intuition for the orientation of the gradient vector.

Figures 9 and 10 compare the frequency domain rendering technique with a conventional spatial domain volume rendering. These images were generated using identical shading and depth cueing. There is no visible difference between the two images.

[3]Assuming each voxel is represented by one byte in the spatial domain algorithm. With shading, spatial domain algorithms require more memory.

[4]A signal whose real part is even and whose imaginary part is odd, i.e. $f(x) = f^*(-x)$.

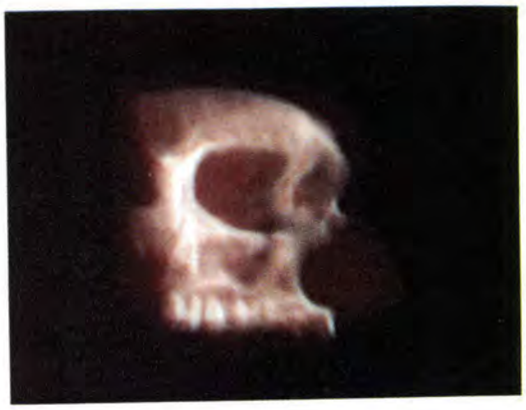


Figure 7: Human head. Frequency domain volume rendering. Data courtesy of North Carolina Memorial Hospital.

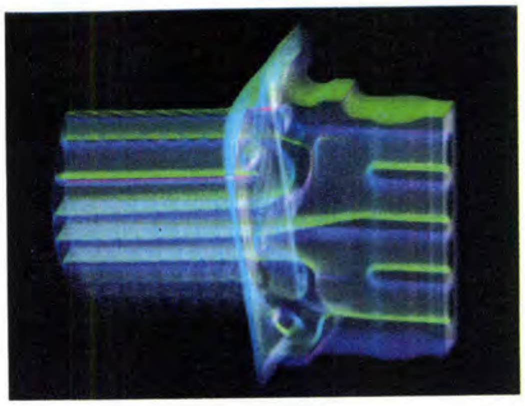


Figure 9: Turbine blade. Frequency domain volume rendering. The blade is lit by a green light (top), a blue light (bottom), and a dim red light (right). Data courtesy of General Electric.

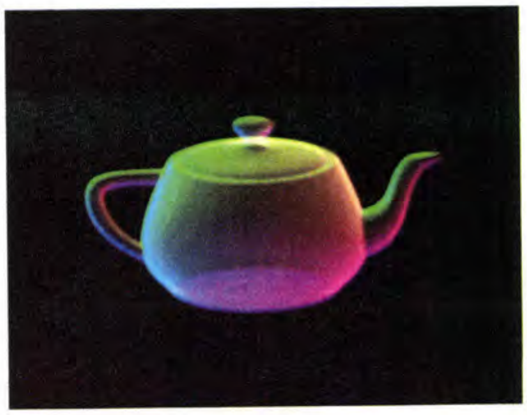


Figure 8: Utah teapot. Frequency domain volume rendering. The pot is lit by a red light (right), a green light (upper left), and a blue light (lower left).

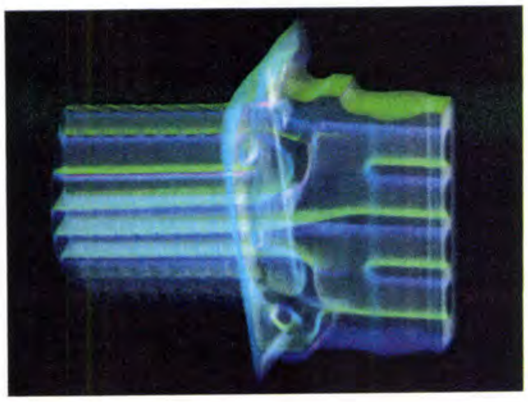


Figure 10: Same dataset as figure 9. Rendered with identical shading and depth cueing but using a spatial domain volume renderer.

The adaptive filtering scheme described in section 4 was implemented using a $3 \times 3 \times 3$ and a $1 \times 1 \times 1$ filter with the maximum difference in response set to ($h_{d\text{-}max}$) 0.3. Figures 7-9 were generated using this scheme. As shown in table 1, the scheme reduced the cost of interpolation to about 15% of the non-adaptive case. Relative error was always below 40dB, a level at which image differences are not visible.

Table 1 also shows rendering times to generate figures 7-9. Rendering times by a spatial domain renderer are also shown for comparison. These times include all necessary operations to create a 2D projection. For the frequency domain rendering technique, it consists of slice extraction (interpolation and resampling), inverse Harteley transform, and format conversion to and from the 16-bit floating point format and the machine's native format. Times were measured on an IRIS Crimson with a 50Mhz R4000 processor using non-optimized code. As the table shows, the running time of the frequency domain method grows much slower than the spatial domain method, which grows at $O(n^3)$.

The effect of round off error caused by the 16-bit floating format was very small. Relative difference from images generated using a 64-bit double precision representation were below 50dB. Figures 7-9 were generated using this format.

7 Conclusions

The use of the Fourier projection slice theorem allows us to replace the $O(n^3)$ spatial domain projection computation that arises in volume rendering with an $O(n^2 \log n)$ frequency domain computation, although the frequency domain projection operator is non-occluding, resulting in a loss of realism. In this paper, we have shown that other $O(n^3)$ spatial domain rendering computations that arise in volume rendering (i.e., shading and depth cueing) can be replaced with $O(n^2)$ frequency domain methods, and we propose that a judicious selection of these methods can restore much of the realism lost by using a non-occluding projection.

The speed advantage of our algorithm over volume rendering is considerable. As our experiments show, a 128^3 volume can be rendered in a fraction of a second on a conventional workstation. Further optimization of the code should achieve interactive rendering without specialized hardware.

Besides its speed advantage, the frequency domain approach lends itself to simple and elegant speed-accuracy tradeoffs. By extracting only the central portion of the 3D spectrum present on a slice, a renderer could provide a low resolution image quickly while the user is rotating the vol-

Volume data	Size	Adaptive filtering			Rendering time	
		Non adaptive	Adaptive			
		Num. ops.†	Num. ops.†	(Ratio)	Freq. domain	Spatial domain
Head	128^3	5.92×10^5	1.01×10^5	(17.1%)	0.54 sec	3.15 sec
Teapot	256^3	1.81×10^6	2.33×10^5	(12.9%)	1.77	24.29
Turbine	256^3	1.85×10^6	3.00×10^5	(16.2%)	2.03	24.38

†A filtering operation consists of a filter table look up, a reference to a voxel, a multiplication, and an addition.

Table 1: Effect of adaptive filtering

ume, to be replaced with a higher quality image when the mouse button or joystick is released.

Since the core computations of the algorithm are convolution and the FFT, an implementation using digital signal processors (DSPs) obviously suggests itself. With the growth of multimedia applications involving video and sound encoding and decoding, such processors are becoming a standard part of most graphics workstations. It should also be noted that these computations exhibit high data level parallelism and can be parallelized in any one of several ways.

With regard to limitations and improvements, further effort should be made to relax the limitations imposed by the linear nature of the Fourier/Hartley transform. The algorithm currently does not allow non-linear attenuation.

Acknowledgements

The authors wish to thank Tom Malzbender for helpful suggestions and his interpolation filter coefficients and Ronald Bracewell for useful hints on the use of the Hartley transform. The notion that shading could be factored with respect to digital compositing, an idea that inspired the present work, was suggested by Brice Tebbs. Discussions with Adam Levinthal were useful in the early stages of this project. Hide Hirase's volume modeling toolkit helped us creating test datasets.

This research was supported by the National Science Foundation (NSF), the National Aeronautics and Space Administration (NASA), and the sponsoring companies of the Stanford Center for Integrated Systems (CIS).

References

[1] Bracewell, Ronald, *The Fourier Transform and its Applications, revised second edition*, McGraw-Hill, 1986.

[2] Bracewell, Ronald, *The Hartley Transform*, Oxford University Press, 1986.

[3] Cohen, Michael and Greenberg, Donald, "The Hemicube: A Radiosity Solution for Complex Environments", *Computer Graphics*, Vol.19, No.3, pp.31-40, 1985.

[4] Drebin, Robert, Carpenter, Loren, and Hanrahan, Pat, "Volume Rendering", *Computer Graphics*, Vol.22, No.4, pp.65-74, 1988.

[5] Dunne, Shane, Napel, Sandy, and Rutt, Brian, "Fast Reprojection of Volume Data", *Proceedings of the First Conference on Visualization in Biochemical Computing*, IEEE Computer Society Press, pp.11-18, 1990.

[6] Hottel, Hoyt, and Sarofim, Adel, "Radiative Transfer", McGraw-Hill, 1967.

[7] Levoy, Marc, "Display of Surfaces from Volume Data", *IEEE Computer Graphics and Applications*, Vol.8, No.3, pp.29-37, 1988.

[8] Levoy, Marc, "Efficient Ray Tracing of Volume Data", *ACM Transactions on Graphics*, Vol.9, No.3, pp.245-261, 1990.

[9] Levoy, Marc, "Volume Rendering using the Fourier Projection-Slice Theorem", *Proceedings of Graphics Interface '92*, Canadian Information Processing Society, pp.61-69, 1992.

[10] Malzbender, Tom, "Fourier Volume Rendering", *ACM Transactions on Graphics*, Vol.12, No.3, July 1993.

[11] Napel, Sandy, Dunne, Shane, and Rutt, Brian, "Fast Fourier Projection for MR Angiography", *Magnetic Resonance in Medicine*, Vol.19, pp.393-405, 1991.

[12] Nishita, Tomoyuki and Nakamae, Eihachiro, "Continuous Tone Representation of Three-Dimensional Objects", *Computer Graphics*, Vol.20, No.4, pp.125-132, 1986.

[13] Pentland, Alex, "Linear Shape from Shading", *International Journal of Computer Vision*, Vol.4, pp.153-162, 1990.

[14] Subramanian, K.R. and Fussel, Donald, "Applying space subdivision techniques to volume rendering", *Proceedings of the First IEEE Conference on Visualization. (Visualization '90)*, IEEE Computer Society Press, pp.150-159, 1990.

[15] Westover, Lee, "Footprint Evaluation for Volume Rendering", *Computer Graphics*, Vol.24, No.4, pp.367-376, 1990.

[16] Zuiderveld, Karel, Koning, Anton, and Viergever, Max, "Acceleration of ray-casting using 3D distance transforms", *Proceedings of the SPIE - Visualization in Biomedical Computing 1992*, Vol.1808, pp.324-335, 1992.

View Interpolation for Image Synthesis

Shenchang Eric Chen, Lance Williams

Apple Computer, Inc.

ABSTRACT

Image-space simplifications have been used to accelerate the calculation of computer graphic images since the dawn of visual simulation. Texture mapping has been used to provide a means by which images may themselves be used as display primitives. The work reported by this paper endeavors to carry this concept to its logical extreme by using interpolated images to portray three-dimensional scenes. The special-effects technique of morphing, which combines interpolation of texture maps and their shape, is applied to computing arbitrary intermediate frames from an array of prestored images. If the images are a structured set of views of a 3D object or scene, intermediate frames derived by morphing can be used to approximate intermediate 3D transformations of the object or scene. Using the view interpolation approach to synthesize 3D scenes has two main advantages. First, the 3D representation of the scene may be replaced with images. Second, the image synthesis time is independent of the scene complexity. The correspondence between images, required for the morphing method, can be predetermined automatically using the range data associated with the images. The method is further accelerated by a quadtree decomposition and a view-independent visible priority. Our experiments have shown that the morphing can be performed at interactive rates on today's high-end personal computers. Potential applications of the method include virtual holograms, a walkthrough in a virtual environment, image-based primitives and incremental rendering. The method also can be used to greatly accelerate the computation of motion blur and soft shadows cast by area light sources.

CR Categories and Subject Descriptors: I.3.3 [Computer Graphics]: Picture/Image Generation; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism.

Additional Keywords: image morphing, interpolation, virtual reality, motion blur, shadow, incremental rendering, real-time display, virtual holography, motion compensation.

1 INTRODUCTION

Generating a large number of images of an environment from closely spaced viewpoints is a very useful capability. A traditional application is a flight in the cabin of an aircraft simulator, whereas the contemporary model is perhaps a walk through a virtual environment; in both cases the same scene is displayed from the view of a virtual camera controlled by the user. The computation of global illumination effects, such as shadows, diffuse and specular inter-reflections, also requires a large number of visibility calculations. A typical approach to this problem is to rely on the computer to repetitively render the scene from different viewpoints. This approach has two major drawbacks. First, real-time rendering of complex scenes is computationally expensive and usually requires specialized graphics hardware. Second, the rendering time is usually not constant and is dependent on the scene complexity. This problem is particularly critical in simulation and virtual reality applications because of the demand for real-time feedback. Since scene complexity is potentially unbounded, the second problem will always exist regardless of the processing power of the computer.

A number of approaches have been proposed to address this problem. Most of these approaches use a preprocess to compute a subset of the scene visible from a specified viewing region[AIRE91, TELL92]. Only the potentially visible objects are processed in the walkthrough time. This approach does not completely solve the problem because there may be viewing regions from which all objects are visible. Greene and Kass[GREE93] developed a method to approximate the visibility at a location from adjacent environment maps. The environment maps are Z-buffered images rendered from a set of discrete viewpoints in 3D space. Each environment map shows a complete view of the scene from a point. An environment map can take the form of a cubic map, computed by rendering a cube of 90° views radiating from that point [GREE86]. The environment maps are pre-computed and stored with viewpoints arranged in a structured way, such as a 3D lattice. An image from a new viewpoint can be generated by re-sampling the environment maps stored in adjacent locations. The re-sampling process involves rendering the pixels in the environment maps as 3D polygons from the new viewpoint. The advantage of this approach is that the rendering time is proportional to the environment map resolutions and is independent of the scene complexity. However, this method requires Z-buffer hardware to render a relatively large number of polygons interactively, a feature still not available on most low-end computers.

This paper presents a fast method for generating intermediate images from images stored at nearby viewpoints. The method has advantages similar to those of Greene and Kass' method. The generation of a new image is independent of the scene complexity. However, instead of drawing every pixel as a 3D polygon, our method uses techniques similar to those used in image morphing[BEIE92]. Adjacent images are "morphed" to create a new image for an in-between viewpoint. The morphing makes use of pre-computed correspondence maps and, therefore, is very efficient. Our experiments with the new method have shown that it can be performed at interactive rates on inexpensive personal computers without specialized hardware.

The new method is based on the observation that a sequence of images from closely spaced viewpoints is highly coherent. Most of the adjacent images in the sequence depict the same objects from slightly different viewpoints. Our method uses the camera's position and orientation and the range data of the images to determine a pixel-by-pixel correspondence between images automatically. The pairwise correspondence between two successive images can be pre-computed and stored as a pair of morph maps. Using these maps, corresponding pixels are interpolated interactively under the user's control to create in-between images.

Pixel correspondence can be established if range data and the camera transformation are available. For synthetic images, range data and the camera transformation are easily obtainable. For natural images, range data can be acquired from a ranging camera [BESL88], computed by photogrammetry [WOLF83], or modeled by a human artist [WILL90]. The camera transformation can be found if the relative positions and orientations of the camera are known.

The idea of using images to represent a virtual environment has been presented previously. An earlier approach uses computer controlled videodiscs to perform surrogate travel [LIPP80]. A more recent approach uses digital movie technologies to construct a virtual museum [MILL92]. In both systems, a user navigates a finite set of routes and directions that have been pre-determined. Our method allows greater flexibility in the navigation because the stored frames can be interpolated smoothly to synthesize arbitrary intermediate points of view.

A static subject or environment portrayed by a restricted set of images indexed by the user's point of view supports a form of "desktop virtual reality" termed "virtual integral holography" [VENO90]. In this context also, our method permits smooth interpolation of the images to present a continuous display sequence, rather than quantizing the user's point of view and jumping to the closest prestored image.

The morphing method can be used to interpolate a number of different parameters, such as camera position, viewing angle, direction of view and hierarchical object transformation. The modeling and viewing transformations can be concatenated to compute the correspondence mapping between two images. Generally, the images can be arranged in an arbitrary graph structure. The nodes of the graph are the images. Each arc in the graph represents a correspondence mapping, which is bi-directional, and two maps are associated with each arc. The number of interpolation parameters determines the dimensionality of the graph. For instance, the graph for a virtual camera moving with two degrees of freedom (the latitudes and longitudes of a sphere bounding an object at a central "look-at" point, for example) is a simple polyhedron (rendering of objects rather than environments will be discussed in more detail in Section 4.4, Image-based Primitives.) The camera's location coordinates index a point on a face of the polyhedron, and the desired view is synthesized by interpolating the images and mappings stored with the vertices and edges of the face. Note that if each image is of the form of an environment map, view angle and direction also can be interpolated by re-projecting the environment map to the desired view orientation [MILL93] without increasing the dimensionality of the graph. Similarly, a camera moving in 3D is supported by a graph which takes the form of a 3D space lattice. The barycentric coordinates of the view location can be used to interpolate among the images attached to the vertices of the enclosing tetrahedron in a lattice of tetrahedra.

For the representation of scenes with objects moving or changes other than those consequent to a change in viewpoint, the graph becomes a general polytope. Generally, arbitrary distortions of surfaces are accommodated by the mapping, as are hierarchical motions of linkages or the limbs of animated characters[1]. To index such an elaborate set of mappings by the various parameters can be an arbitrarily complex process, requiring multivariate interpolation of a multidimensional graph.

Without loss of generality, this paper will concentrate on the interpolation of the camera position in 1D and 2D space (accommodating "virtual holograms" of objects as well as restricted navigation in 3D scenes). The scene is assumed to be static, and all the image changes are as a result of camera movement. Although the method can be applied to natural images, only synthetic ones have been attempted in the work described here. Interpolation of images accurately supports only view-independent shading. Reflection mapping or Phong specular reflection could be performed with separate maps for reflection map coordinates or normal components, but only diffuse reflection and texture mapping have been presented here.

Section 2 introduces the basic algorithms of the method as well as its limitations and optimizations. Section 3 gives implementation details and shows some examples. Section 4 shows applications of the method to virtual reality, temporal anti-aliasing, generating shadows from area lights, image-based display primitives and incremental rendering ("progressive refinement"). Conclusions and future directions are discussed in the last section.

2 VISIBILITY MORPHING

Image morphing is the simultaneous interpolation of shape and texture. The technique generally involves two steps. The first step establishes the correspondence between two images and is the most difficult part of most morphing methods. The correspondence is usually established by a human animator. The user might, for example, define a set of corresponding points or line segments within a pair or set of images. An algorithm is then employed to determine the correspondence (mapping) for the remainder of the images[BEIE92]. The second step in the process is to use the mapping to interpolate the shape of each image toward the other, according to the particular intermediate image to be synthesized, and to blend the pixel values of the two warped images by the same respective coefficients, completing the morph.

Our method uses the camera transformation and image range data to automatically determine the correspondence between two or more images. The correspondence is in the form of a "forward mapping." The mapping describes the pixel-by-pixel correspondence from the source to the destination image. The mapping is also bi-directional since each of the two images can act as the source and the destination. In the basic method, the corresponding pixels' 3D screen coordinates are interpolated and the pixels from the source image are moved to their interpolated locations to create an interpolated image. For pixels which map to the same pixel in the interpolated image, their Z-coordinates are compared to resolve visibility. Cross-dissolving the overlapping pixels' colors may be necessary if the image colors are not view-independent. This process is repeated for each of the source images.

This method is made more efficient by the following two properties. First, since neighboring pixels tend to move together in the mapping, a quadtree block compression is employed to exploit this coherence. Adjacent pixels which move in a similar manner are grouped in blocks and moved at the same time. This compression is particularly advantageous since a view-independent visible priority among the pixel blocks can be established. The pixel blocks are sorted once by their Z-co-

[1]Establishing such elaborate mappings is straightforward for synthetic images, a classic vision problem for natural ones.

ordinates, when the maps are created, and subsequently displayed from back to front to eliminate the overhead of a Z-buffer for visibility determination.

We will describe our method in terms of the morphing between two images first. Generalization of the method to more images is straightforward and will be discussed later.

2.1 Establishing Pixel Correspondence

As a camera moves, objects in its field of view move in the opposite direction. The speed of each object's apparent movement is dependent on the object's location relative to the camera. Since each pixel's screen coordinates (x, y and z) and the camera's relative location are known, a 4x4 matrix transformation establishes a correspondence between the pixels in each pair of images. The transformations can be pre-computed and reduced to a 3D spatial offset vector for each of the pixels. The offset vector indicates the amount each of the pixels moves in its screen space as a result of the camera's movement. The offset vectors are stored in a "morph map," which represents the forward mapping from one image to another. This map is similar in concept to a disparity map computed from a stereo pair[GOSH89], the field of offset vectors computed for "optical flow" analysis[NAGE86], or motion compensation in video compression and format conversion[MPEG90]. For a computed image or range image, an exact pixel-by-pixel map can be created. The mapping is many-to-one because many pixels from the first image may move to the same pixel in the second image. Therefore, the morph map is directional and two morph maps are needed for a pair of images.

The use of a pre-computed spatial look-up table for image warping has been presented in [WOLB89]. Wolberg used the look-up table to implement arbitrary forward mapping functions for image warping. Wolberg's maps contained absolute coordinates rather than offset vectors.

In a typical image morph, as described in the beginning of this section, a sparse correspondence provided by a human operator is used to perform strictly two-dimensional shape interpolation. Such a morph can also be used to interpolate stored images in order to represent 3D scenes or objects, as suggested in [POGG91]. The advantages of our method are that the correspondence is dense (every pixel has an explicitly computed map coordinate), the correspondence is automatic (rather than relying on human effort), and the explicit prestored maps permit the image deformations to be generated very quickly.

2.2 Interpolating Correspondences

To generate an in-between view of a pair of images, the offset vectors are interpolated linearly and the pixels in the source image are moved by the interpolated vector to their destinations. Figure 1 shows the offset vectors, sampled at twenty-pixel intervals, for the camera motion sequence in Figure 3.

The interpolation is an approximation to the transformation of the pixel coordinates by a perspective viewing matrix. A method which approximates the perspective changes with local frame shifting and scaling is presented in [HOFM88]. Perspective transformation requires multiplication of the pixel coordinates by a 4x4 matrix and division by the homogeneous coordinates, a rather computationally taxing process, although bounded by image resolution rather than scene complexity. Linear interpolation of pixel coordinates using the morph maps, on the other hand, is very efficient and can be performed incrementally using forward differencing.

If the viewpoint offset is small, the interpolation is very close to the exact solution. Moreover, quadratic or cubic interpolation, though slightly more expensive to perform, can be used to improve the accuracy of the approximation. When the viewpoint moves parallel to the viewing plane, the linear interpolation produces an exact solution. This case is demonstrated in Figure 2a, which traces the paths of mapped pixels in the interpolated image as the viewpoint traverses the four corners of a square parallel to the viewing plane. The squares in the figure are the extents of the pixel movement. Because the squares are parallel to the viewing plane, the linear interpolation of the square corners produces the same result as perspective transformation. Another special case is when the viewpoint moves perpendicular to the viewing plane along a square parallel to the ground(Figure 2b). The resulting pixel locations form trapezoids, which are the projections of squares parallel to the ground. The trapezoids can be interpolated linearly in the horizontal direction. The vertical direction requires perspective divisions. The divisions can be avoided if a look-up table indexed by the vertical offset is pre-computed for each possible integer height of the trapezoids. The second case can be generalized to include the case when the squares are perpendicular to both the ground and the viewing plane. If the viewpoints are aligned with a 3D lattice, the result will always fall into one of the above two cases, which allows us to use linear interpolation to generate an exact solution.

2.3 Compositing Images

The key problem with forward mapping is that overlaps and holes may occur in the interpolated image.

2.3.1 Overlaps

One reason overlaps occur is due to local image contraction. Local image contraction occurs when several samples in a local neighborhood of the source image move to the same pixel in the interpolated image. A typical example of this case is when our view of a plane moves from perpendicular to oblique. Perspective projection causes the image to contract as the plane moves away from the point of view. In the mapping, the samples on the far side of the plane contract while the samples on the near side expand. Contraction causes the samples to overlap in the target pixels.

Multiple layers of pixel depths also will cause the samples to overlap, as in the case of the foreground sculpture in Figure 3. Resolving this case is really a hidden surface problem. One way of solving this problem is to use the Z-buffer algorithm to determine the frontmost pixel. A more efficient way of determining the nearest pixel is presented in the Optimization Section.

2.3.2 Holes

Holes between samples in the interpolated image may arise from local image expansion when mapping the source image to the destination image. This case is shown in Figure 3 where a source image is viewed from viewpoints rotated to the right. The cyan regions indicate holes. Generally, a square pixel in the source image will map to a quadrilateral in the destination image. If we interpolate the four corners of the square instead of the pixel's center, the holes can be eliminated by filling and filtering the pixels in the destination quadrilateral.

A more efficient, though less accurate, method to fill the holes is to interpolate the adjacent pixels' colors or offset vectors. The holes are identified by filling the interpolated image with a reserved "background" color first. For those pixels which still retain the background color after the source to target mapping, new colors are computed by interpolating the colors of adjacent non-background pixels. Alternatively, we can interpolate the offset vectors of the adjacent pixels. The interpolated offset is used to index back to the source image to obtain the new sample color. Note that using a distinguished background color may not identify all the holes. Some of the holes may be created by a foreground object and are filled by a background object behind it (e.g., the holes in the sculpture in the rightmost image in Figure 3). This problem is alleviated, though not completely eliminated, when more source images are added as described below (e.g. Figure 5d).

Holes may also arise from sample locations invisible in each of the source images but visible in the interpolated image. The hole region, as shown in Figure 4, is the intersection of the umbra regions cast by viewpoints A and B and the visible region from point M. The small circle in the hole region is completely missed by the two source images from points A and B. One way of solving this problem is to use multiple source images to minimize the umbra region. Figure 5a shows the holes (cyan pixels) created by rotating one source image. Figure 5b shows that the number of holes is significantly less when two sources images are used. The number of holes can be reduced further if we place the two source viewpoints closer (Figure 5c). The remaining holes can be filled by interpolating the adjacent pixels(Figure 5d). If the images are computer-generated, a ray-tracing type of rendering can be used to render only those missing pixels.

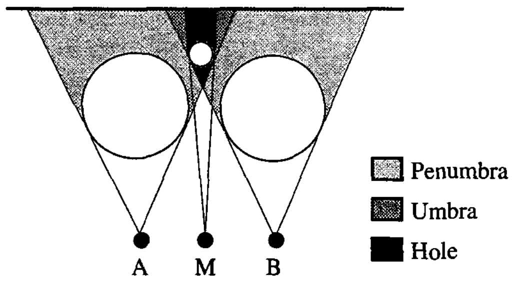


Fig. 4 Penumbra, umbra and hole regions

2.4 Optimization

The basic method is made more efficient by the following two steps.

2.4.1 Block Compression

Since adjacent pixels tend to move together in the mapping, a block compression scheme such as a quadtree can be applied to compress the morph map. The compression serves two purposes. First, it reduces the size of the morph map. Second, it allows us to interpolate offsets for entire blocks instead of pixel-by-pixel. The second aspect greatly accelerates the interpolation process as the main cost in the process is the interpolation of the offset vectors.

The compression ratio is related to the image depth complexity and the viewpoint movement. For images with high depth complexity, the compression ratio is usually low. The ratio is also lower if the viewpoint's movement results in greater pixel depth change. Figure 6 shows the quadtree decomposition of the morph map for the image sequence in Figure 3. The maximal offset threshold within a block is one pixel in Figure 6a and two pixels in Figure 6c, which means the offset vector coordinates within a block do not differ more than one or two pixel units. The compression ratio in Figure 6a is 15 to 1 and in Figure 6b is 29 to 1 (i.e., the number of blocks vs. the number of pixels).

The threshold provides a smooth quality degradation path for increased performance. Large threshold factors result in fewer quadtree blocks and, therefore, reduce the interpolation time. The performance gain is at the expense of increasing blockiness in the interpolated image. The interpolation times in Figure 6b and 6d are accelerated by a factor of 6 and 7 respectively. Note that the speedup factor does not grow linearly with the compression ratio because the same number of pixels still need to be moved.

2.4.2 View-Independent Visible Priority

In the basic method, the Z-buffer algorithm is used to resolve visibility. However, as shown in Figure 7, the A-closer-than-B priority established in View1 is still valid in View2, since Point A and Point B do not overlap in View2. The priority is incorrect in View3 when A and B overlap. As long as the angle θ in the figure is less than 90 degrees, the A-B priority does not need to be changed when the viewpoint is moved. This observation allows us to establish a view-independent visible priority for every source pixel for a viewing range. The pixels are ordered from back to front based on their original Z-coordinates when the morph maps are created, and are subsequently drawn in a back-to-front order in the interpolation process. This ordering of the samples, or sample blocks, eliminates the need for interpolating the Z-coordinates of every pixel and updating a Z-buffer in the interpolation process.

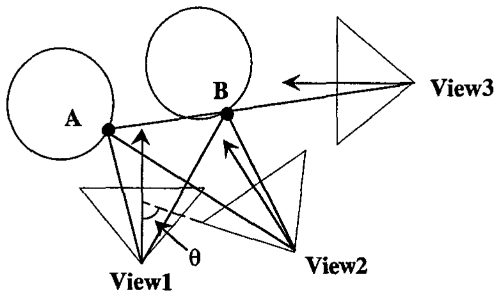


Fig. 7 View-independent visible priority

Note that the priority established here is for image pixels rather than for the underlying objects, unlike list-priority algorithms for hidden-surface removal[SCHU69].

This method applies to multiple source images as well. The source images' pixel Z-coordinates are transformed to a single coordinate system for establishing the Z-priority. All the pixels in the source images are sorted into the same priority list.

The priority can be assigned to every quadtree pixel block. With static objects and a moving camera, pixel offsets are directly related to Z-coordinates. Since the pixels within a block have similar offsets, they also have similar Z-coordinates. The Z-coordinates within a block are filtered to determine a Z value for the priority sort. The result is a sorted list of pixel blocks valid for the entire range between views.

3 IMPLEMENTATIONS

The method presented above can be summarized as follows.

3.1 Preprocessing

The preprocessing stage establishes the correspondence between each pair of source and destination images. As mentioned in Section 1, the source images are connected to form a graph structure. Each node of the graph contains a source image, its range data and camera parameters (i.e., camera's position, orientation). For each set of adjacent nodes in the graph, a sorted list of quadtree blocks is created (e.g., a block list is created for every triangle in a 2D lattice structure). Each block in the list contains a pointer to a pixel block in a source image, the size, the screen coordinates and the offset vectors of the block. The block list is created in the following steps:

Step 1. Get input data: a source node (image, range data and camera parameters), a destination node (only the camera parameters are needed) and a threshold factor for the quadtree decomposition.

Step 2. Create a morph map from the source to the destination (Section 2.1).

Step 3. Decompose the morph map into quadtree blocks and add the blocks to a block list (Section 2.4.1).

Step 4. Repeat Step 1 to 3 for each directional arc connecting the set of nodes.

5. Sort the block list from back to front by the blocks' Z-coordinates.

3.2 Interactive Interpolation

In the interactive interpolation stage, the block list corresponding to a new viewing location is retrieved. The parametric coordinates of the location with respect to the adjacent nodes are used as interpolation parameters. An interpolated image for the new location is generated in the following steps:

Step 1. Get input data: interpolation parameters and a sorted block list.

Step 2. Fill the interpolated image with a distinguished background color.

Step 3. For every block in the list in back-to-front order, compute its new location from the offset vectors and the interpolation parameters. Copy the pixel block from the source image to its new location in the interpolated image (Section 2.2).

Step 4. For every pixel in the interpolated image that still retains the background color, compute its color by filtering the colors of the adjacent non-background pixels (Section 2.3.2).

3.3 Examples

Figure 8 shows a sequence of images generated by moving the viewpoint to the right. The images were rendered at 256x256 resolution using progressive radiosity [COHE88] from a model created for the Virtual Museum project[MILL92].

Figure 9 shows two intermediate images created by morphing the leftmost and rightmost images. Each image took 0.17 second to generate (excluding the preprocessing time) on a Macintosh Quadra 950.

Note that for the interpolation to work properly, the source image cannot be anti-aliased. Anti-aliasing is view-dependent. It blends silhouette pixel colors from a particular viewpoint. Since the Z-buffer cannot be anti-aliased in the same way, the anti-aliased silhouette pixels may attach to either the foreground or the background objects depending on the quantization of the Z-buffer. This problem can be solved by morphing high-resolution unfiltered source images and then filtering the interpolated image.

The method can be applied to interpolating more than two source images. Figure 10 shows a sequence of images interpolated from the four source images in the corners. The viewpoints of the source images form a square parallel to the viewing plane. Therefore, as discussed before, linear interpolation is an exact solution to the perspective transformation. New images are computed from the nearest three corner images. The barycentric coordinates of the new viewpoint are used to interpolate the three images. Dividing the lattice into simplices minimizes the cost of interpolation.

4 APPLICATIONS

The morphing method can be used in a wide variety of applications which require fast visibility computations of a predefined static scene. Simulation and virtual reality applications typically require a scene to be displayed interactively from different viewpoints. Temporal anti-aliasing, or motion blur, can be accelerated by using morph maps to integrate image samples over time. The image samples are interpolated from key images using the morphing method. We also present an application of morph mapping to compute shadows from area lights using the shadow buffer method [WILL78]. The morphing method makes it possible to define a new class of graphic display primitives based on images. This approach is also useful in incremental rendering as it provides a way to reuse the pixels computed for previous images.

4.1 Virtual Reality

Instead of representing a virtual environment as a list of 3D geometric entities, the morphing method uses images (environment maps). To perform a walkthrough, the images adjacent to the viewpoint are interpolated to create the desired view.

In addition to supporting walkthroughs in virtual environments, the method can be used to create virtual holograms, where the display on the screen will change with respect to the user's viewpoint to provide 3D motion parallax. One existing approach uses 3D rendering to display the scene from the viewpoint obtained by a head location sensor[DEER92]. Another approach uses a finite set of pre-rendered frames, each corresponding to a particular viewing location[VENO90]. With the morphing method, only a few key images are required. The interpolation can generate the in-between frames. Figure 10 shows a sequence of images with vertical and horizontal motion parallax.

The image-based morphing method is inexpensive computationally and provides a smooth quality-speed tradeoff. Although the total storage requirement may be large, the amount of data needed to compute a frame is relatively small and can be read from secondary storage as needed. This approach is very appropriate for CD-ROM based devices because of their large storage capability. As the complexity of geometrical models increases, the advantage of image-based approaches will be more significant because of their bounded overhead.

Another advantage of using the image-based approach is that a real environment can be digitized by photographic means. Using a camera to capture the environment usually is much easier than modeling it geometrically. Although our method relies on range data to establish the correspondence between images, range data should be easier to obtain than the complete 3D geometry of the environment.

4.2 Motion Blur

If an image in a motion sequence is a sample at an instant of time instead of over a time interval, the motion will appear to be jerky and the image is said to be aliased in the temporal domain. One way to perform temporal anti-aliasing is super-sampling. The motion is sampled at a higher rate in the temporal domain and then the samples are filtered to the displayed rate. Super-sampling requires the computation of many more samples. For images which are expensive to render, this technique is very inefficient.

The morphing method allows additional temporal samples to be created by interpolation. The interpolation time is constant regardless of the rendering time for each frame. The sampling rate is determined by the largest offset vector from the morph map in order to perform proper anti-aliasing. Figure 11a is a motion blurred image computed from 32 source images for the camera motion in Figure 8. The images were first rendered at 512x512 resolution and then filtered down to 256x256 resolution before temporal anti-aliasing was performed. The temporal samples were anti-aliased with a box filter. Each image took around 5 seconds to render on a high-end workstation with 3D graphics hardware support. Figure 11b was computed from the same number of images interpolated from three of the source images. Each interpolated image took 0.6 second to compute on a Macintosh Quadra950. The only minor visible difference between the two images is the top of the inside loop of the foreground sculpture, due to the holes created from the interpolation as discussed previously.

The super-sampling approach requires the sampling rate to be determined based on the worst case. For images with fast moving objects and slowly moving backgrounds, this method is not very efficient. One way to solve this problem is to segment the images based on object movement and use different sampling rates for each segment. For instance, the foreground sculpture in this figure needs to be sampled at the highest rate while the wall behind it needs only a few samples. In the case of motion caused by viewpoint changes as in this figure, the segments can be sorted in order of depth as discussed in Section 2.4.2. Each segment is filtered independently and a temporal coverage value for each pixel is kept to indicate the ratio of background samples vs. all samples. The multiple segment layers are then composited in front-to-back order with each segment's pixel colors attenuated by the coverage value from the previous segment.

4.3 Shadows

A very general and efficient way of rendering shadows is the shadow buffer algorithm [WILL78]. The algorithm computes a Z-buffer (i.e., shadow map) from the point of view of the light source. To compute shadows, a surface point's coordinates are transformed to the light source's space and its Z-coordinate is compared to the corresponding Z-coordinate in the shadow map. If the point is further away then it is in shadow.

The algorithm only works for point light sources. To approximate a linear or an area source, many point lights may be needed [SHAP84]. The cost of computing the shadows is proportional to the number of point sources used.

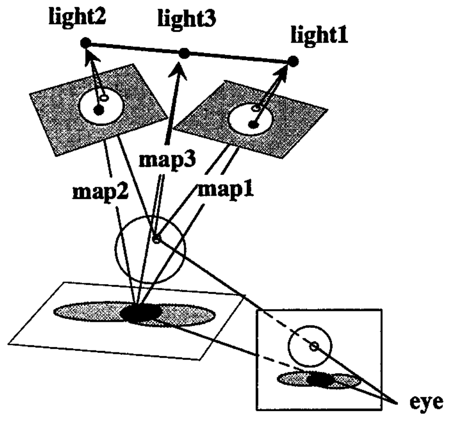


Fig. 12 Shadow buffer interpolation for a linear light source

The morphing method can be used to significantly reduce the cost of computing the shadow map for each of the point sources. Figure 12 illustrates the process of using the method to compute shadows from a linear light source. A shadow map is computed first for each of the two end points of the source (i.e., light1 and light2) using the conventional rendering method. A morph map from the viewpoint to each of the two end points is also computed to transform the screen coordinates to each point source's coordinate space (i.e., map1 and map2). The shadow map for an in-between point (e.g., light3) on the linear source is interpolated from the corner shadow maps using the morphing method. The same interpolation factor is used to interpolate the two morph maps (map1 and map2) to create a morph map from the viewpoint to the in-between light source point (map3). The standard shadow buffer algorithm is then used to compute shadows for the in-between point source. The process is repeated for all the in-between points at a desired interval. The resulting shadow images are composited to create the soft shadow of the linear source. This method can be generalized to any area or volume light source.

Figure 13 shows the result after compositing 100 in-between shadow images generated by randomly distributed points on a rectangular light source above the triangle. Four source shadow maps located at the corners of the rectangle were created for the interpolation. The shadow maps were rendered at 512x512 resolution and the shadow image resolution is 256x256. Percentage closer filtering [REEV87] was used to anti-alias the shadows for each image. Each shadow image took 1.5 seconds to compute. Shading for the illuminated pixels was computed by Lambert's Law weighted by the projected size of the rectangle source over the pixel.

4.4 Image-Based Primitives

A 3D object is perceived on a flat display screen through a series of 2D images. As long as we can generate the images from any viewpoint, it does not matter if a 3D description of the object is available. The morphing method permits any view of an object to be generated by interpolation from some key images. Therefore, a new class of primitives based on images can be defined. These image-based primitives are particularly useful for defining objects of very high complexity since the interpolation time is independent of the object complexity.

Figure 14 shows a sequence of images of a rotating teapot generated by the morphing method. The middle images were generated by interpolating the two key images at the extreme left and right. The key images were rendered with viewpoints rotated 22.5 degrees around the center of the teapot. A larger angular increment of the key images may result in holes and distortions as a result of the linear interpolation. Figure 15 is the same source images extrapolated to show the pixel blocks which compose the teapot.

Rendering an object using the morphing method is really not different from rendering a complete scene as described previously. The image-based object or scene can be treated as a "sprite" that can be composited with images generated by other means.

4.5 Incremental Rendering

Adjacent images in an animation sequence usually are highly coherent. Therefore, it's desirable to perform the rendering incrementally. Ideally, the rendering should be limited to only the pixels which are different from the previous frame. However, searching for the pixels that change is not always trivial. Some incremental rendering approaches which make use of frame-to-frame coherence were presented in [CHEN90], [JEVA92].

The morphing method provides a natural way of making use of frame coherence. For an animation sequence where the motion of every frame is known in advance, the frames can be rendered initially at a coarse temporal sampling rate. The remaining frames can then be computed by the morphing method. The missing samples or view-dependent shading, such as highlights, of the interpolated frames can be computed by additional rendering. If accuracy rather than speed is the main concern, the map-based interpolation or extrapolation of pixel coordinates can be replaced by perspective transformation.

5 CONCLUSIONS AND FUTURE DIRECTIONS

The interactive speed which the image-based display has achieved on modest computing platforms has fulfilled our primary goal in pursuing this research. In addition to this primary objective, we have demonstrated effective application of the view interpolation approach to computing some of the more complex rendering effects. Image-based computer graphics promises to be a productive area of research for some time. A number of intriguing research problems suggest themselves:

An automatic camera has been developed to record an array of images of an object from viewpoints surrounding it [APPL92]. What are the prospects for automatic camera location selection to minimize the number of holes in the interpolated images? Similarly, what are good algorithmic criteria for dispensing with as many recorded images as possible, or selecting the best subset of images to represent the object?

By modeling the 3D transformation from one image to the next by a field of straight-line offsets, we introduce an approximation analogous to polygonization (except in the restricted cases mentioned in Section 2.2). Higher-dimensional, rather than linear, interpolation might be expected to better approximate the arcs traversed by objects rotating between views. Curved motion blur is another possible benefit of higher-order interpolation.

View-dependent shading such as specular reflection would extend the useful range of morphing as a display technique. One possibility mentioned previously is to define additional maps for specular surfaces, which specify normal components or reflection map coordinates.

Special-purpose image compression might profit greatly from morph-mapping algorithms. The resemblance of the morph maps to motion-compensation vectors commonly used in video sequence compression has been mentioned. These vectors, used in format conversion to address the interlace problem, and in compression to squeeze a little more redundancy out of the signal, also find application in optical flow algorithms for tracking objects in the visual field. The redundancy removed from the video sequence by motion compensation is limited, as it applies only between successive frames. In a morph mapping encoder, objects which appear and disappear repeatedly could be encoded with a small set of maps. The decoder, a hybrid of an image warper and a graphics pipeline, would use them as "sprites" from a catalog of maps.

The representation of objects and surfaces as sets of images and maps, possibly pyramidal maps, suggests the application of morph mapping to more general global illumination models. The approach of determining visibility to an area light source to compute soft shadows can be extended to treating all surfaces as sources of radiosity. For many global illumination problems, a few images and morph maps can serve to represent hundreds or thousands of computed images.

6. ACKNOWLEDGMENTS

Thanks to the Virtual Museum team for the museum model and images. Dan Venolia anticipated the use of range images as display primitives (without interpolation) in his virtual holography work. Ken Turkowski contributed the teapot images. Ned Greene, Nelson Max and members of the Advanced Technology Computer Graphics Group have offered useful ideas and criticism. Frank Crow and Apple Computer's continuous support of this research is highly appreciated.

REFERENCES

[AIRE91] Airey, J., J. Rohlf and F. Brooks. Towards Image Realism with Interactive Update Rates in Complex Building Environments. ACM SIGGRAPH Special Issue on 1990 Symposium on Interactive 3D Graphics, 41-50.

[APPL92] Apple Human Interface Group. Object Maker. [exhibit] In Interactive Experience, CHI'92, Monterey CA.

[BESL88] Besl, P.J. Active Optical Range Imaging Sensors. Machine Vision and Applications Vol. 1, 1988, 127-152.

[BEIE92] Beier, T. and S. Neely. Feature-Based Image Metamorphosis. SIGGRAPH'92 Proceedings, 35-42.

[CHEN90] Chen, S. E. Incremental Radiosity: An Extension of Progressive Radiosity to an Interactive Image Synthesis System. SIGGRAPH'90 Proceedings, 135-144.

[COHE88] Cohen, M. F., S. E. Chen, J. R. Wallace and D. P. Greenberg. A Progressive Refinement Approach to Fast Radiosity Image Generation. SIGGRAPH'88 Proceedings, 75-84.

[DEER92] Deering, M. High Resolution Virtual Reality. SIGGRAPH'92 Proceedings, 195-202, 1992.

[GOSH89] Goshtasby, A. Stereo Correspondence by Selective Search. Proc. Japan Computer Vision Conf., 1-10, July, 1989.

[GREE86] Greene, N. Environment Mapping and Other Applications of World Projections. IEEE CG&A, Vol. 6, No. 11, November, 1986.

[GREE93] Greene, N. and M. Kass. Approximating Visibility with Environment Maps. Technical Report 41, 1993, Apple Computer, Inc.

[HOFM88] Hofman, G. R. The Calculus of the Non-Exact Perspective Projection. Eurographics'88 Proceedings, 429-442

[JEVA92] Jevans, D. Object Space Temporal Coherence for Ray Tracing. Graphics Interface'92 Proceedings, 176-183, 1992.

[LIPP80] Lippman, A. Movie Maps: An Application of the Optical Videodisc to Computer Graphics. SIGGRAPH'80 Proceedings, 32-43.

[MILL92] Miller, G., E. Hoffert, S. E. Chen, E. Patterson, D. Blacketter, S. Rubin, S. A. Applin, D. Yim and J. Hanan. The Virtual Museum: Interactive 3D Navigation of a Multimedia Database. The Journal of Visualization and Computer Animation, Vol. 3, No. 3, 183-198, 1992.

[MILL93] Miller, G.and S. E. Chen. Real-Time Display of Surroundings Using Environment Maps. Technical Report 42, 1993, Apple Computer, Inc.

[MPEG90] MPEG Video Committee Draft, December, 1990.

[NAGE86] Nagel, H.-H. Image Sequences - Ten (octal) Years from Phenomenology to a Theoretical Foundation. Proc. 8th ICPR, Paris 1986, 1174-1185.

[POGG91] Poggio, T. and R. Brunelli. A Novel Approach to Graphics. MIT A.I. Memo No. 1354, C.B.I.P. Paper No. 71, February, 1992.

[REEV87] Reeves, W. T., D. H. Salesin and R. L. Cook. Rendering Antialiased Shadows with Depth Maps. SIGGRAPH'87 Proceedings, 283-291.

[SCHU69] Schumacker, R., B. Brand, M. Gilliland, and W. Sharp. Study for Applying Computer-Generated Images to Visual Simulation, Technical Report AFHRL-TR-69-14, NTIS AD700375, U.S. Air Force Human Resources Lab., Air Force Systems Command, Brooks AFB, TX, September, 1969.

[SHAP84] Shapiro, B. L., N. I. Badler. Generating Soft Shadows with a Depth Buffer Algorithm. IEEE CG&A, Vol. 4, No. 10, 5-38, 1984.

[TELL92] Teller, S and C. Sequin. Visibility Preprocessing for Interactive Walkthroughs. SIGGRAPH'91 Proceedings, pp.61-69, 1991.

[VENO90] Venolia, D. and L. Williams. Virtual Integral Holography. Proc. SPIE-Extracting Meaning from Complex Data: Processing, Display, Interaction (Santa Clara, CA, February, 1990), 99-105.

[WILL78] Williams, L. Casting Curved Shadows on Curved Surfaces. SIGGRAPH'78 Proceedings, 270-274.

[WILL90] Williams, L. 3D Paint. ACM SIGGRAPH Special Issue on 1990 Symposium on Interactive 3D Graphics, 225-233.

[WOLB89] Wolberg, G. and T. E. Boult. Separable Image Warping with Spatial Lookup Tables. SIGGRAPH'89 Proceedings, 369-377.

[WOLF83] Wolf, P. R. Elements of Photogrammetry, McGraw-Hill, New York, 1983.

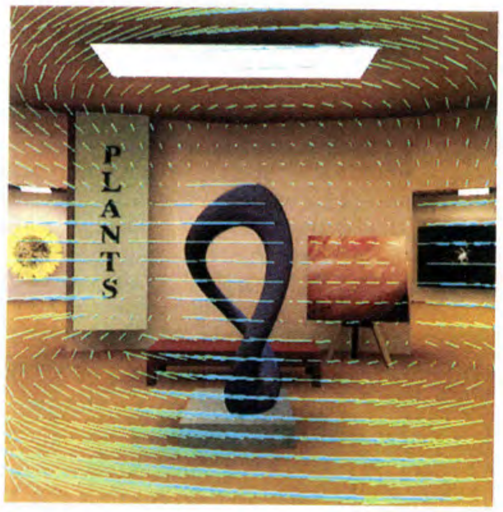
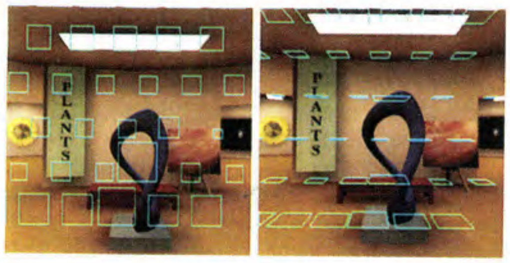


Fig. 1 *Offset vectors for the camera motion in Figure 3.*



(a) *(b)*

Fig. 2 *Extents of pixel movement for 2D viewpoint motions: a) viewpoints parallel to the viewing plane, b) viewpoints parallel to the ground. (Source pixels are in the lower right corner of each extent.)*

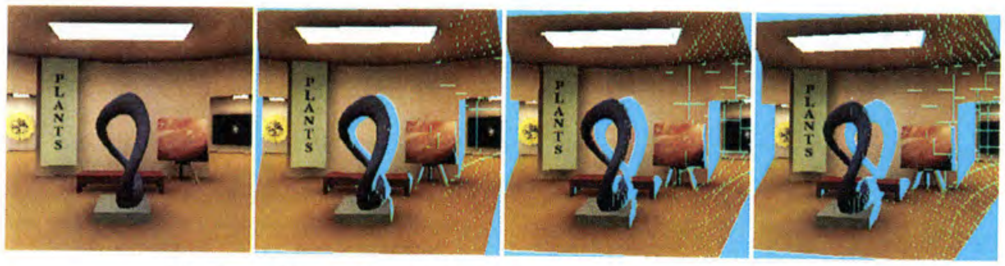
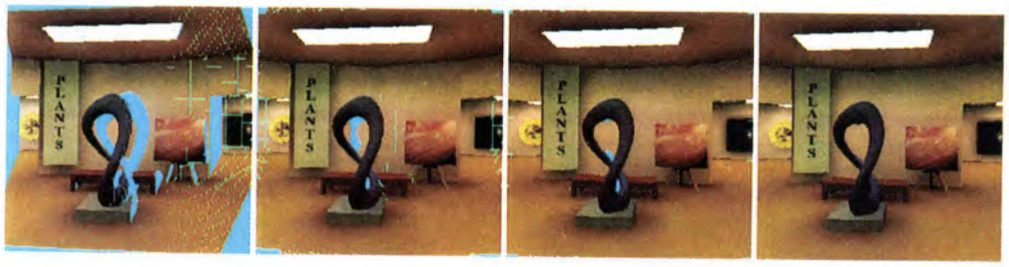
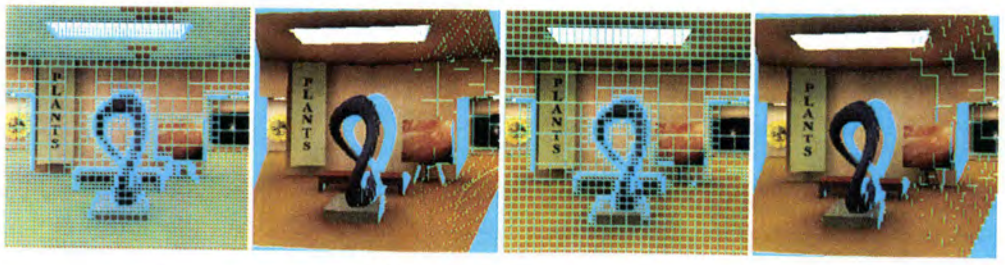


Fig. 3 *A source image viewed from a camera rotated to the right.*



(a) *(b)* *(c)* *(d)*

Fig. 5 *(a) Holes from one source image, (b) holes from two source images, (c) holes from two closely spaced source images, (d) filling the holes with interpolation.*



(a) *(b)* *(c)* *(d)*

Fig. 6 *Quadtree decompositions of a morph map: (a) compression ratio: 15 to 1, speedup factor: 6; (b) interpolated image from (a); (c) compression ratio: 29 to 1, speedup factor: 7; (d) interpolated image from (c).*

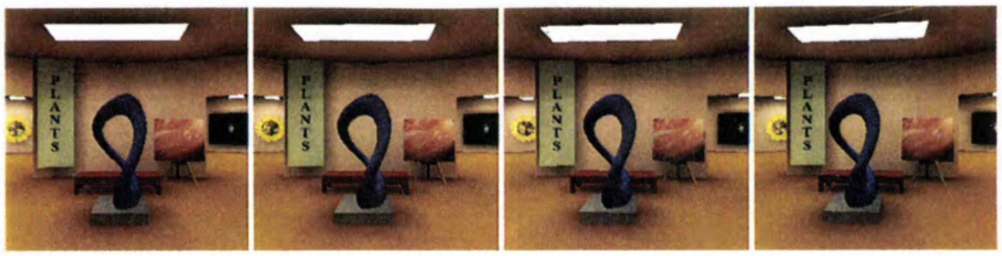


Fig. 8 Rendered Virtual Museum images.

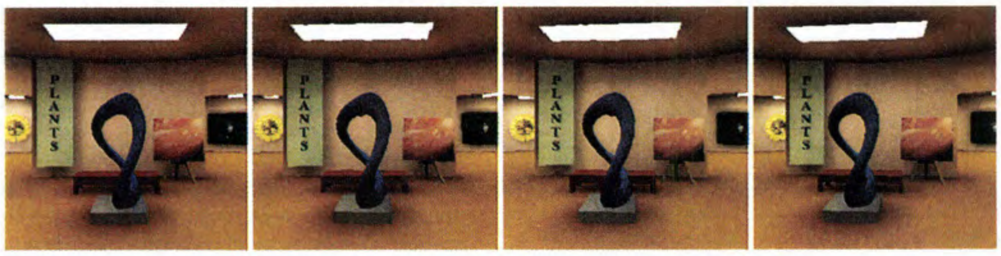


Fig. 9 Interpolated Virtual Museum images (two middle ones).

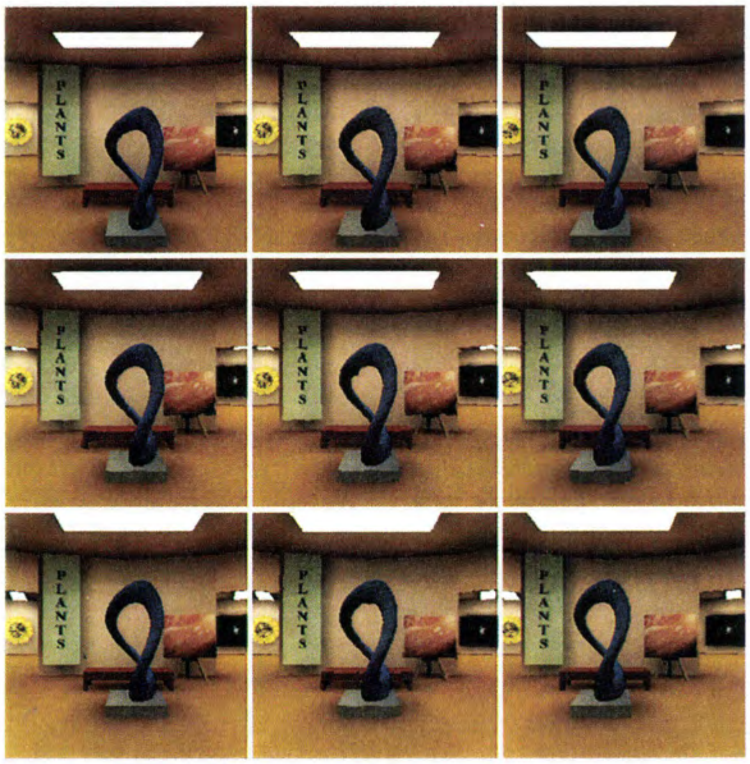
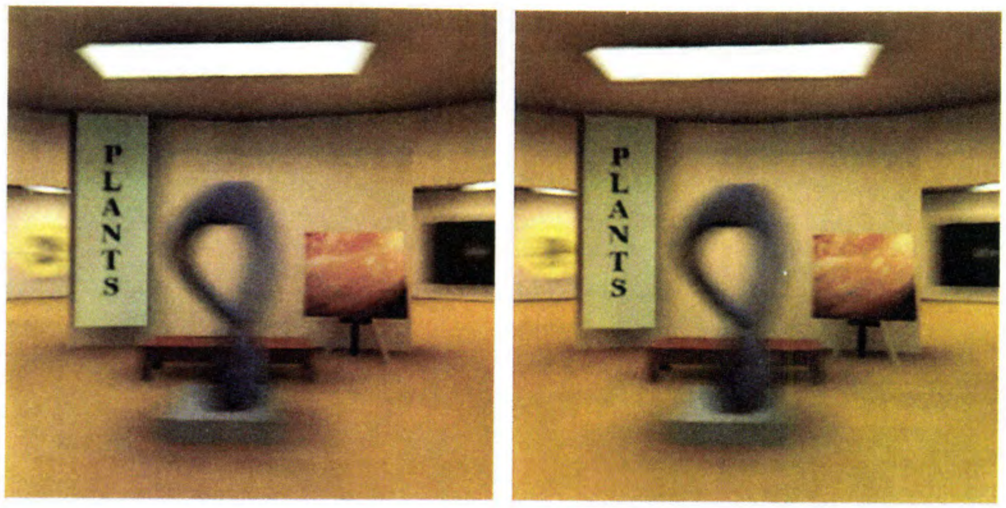


Fig. 10 2D interpolation. The source images are in the corners. All the other images are interpolated from their nearest three source images. (The center one is interpolated from the upper two and the lower left corners.)



(a) *(b)*

Fig.11 (a) Motion blur computed from source images, (b) motion blur computed from interpolated images[2].

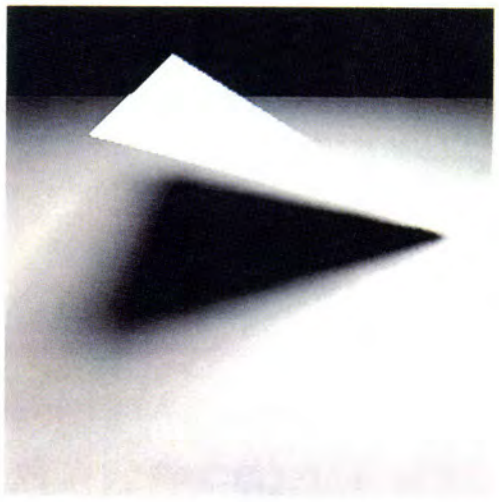


Fig.13 Shadow from a rectangular area light computed with the shadow map interpolation.

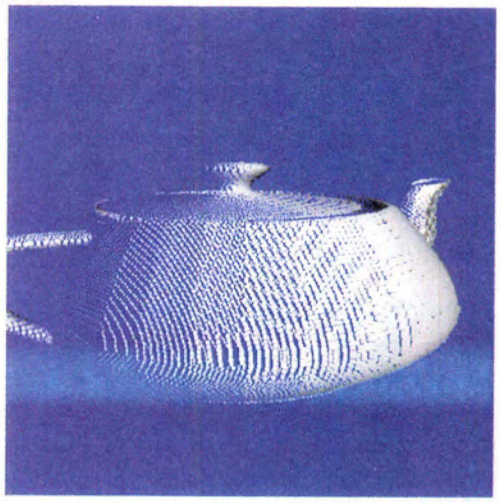


Fig.15 Teapot extrapolated to show the quadtree pixel blocks.

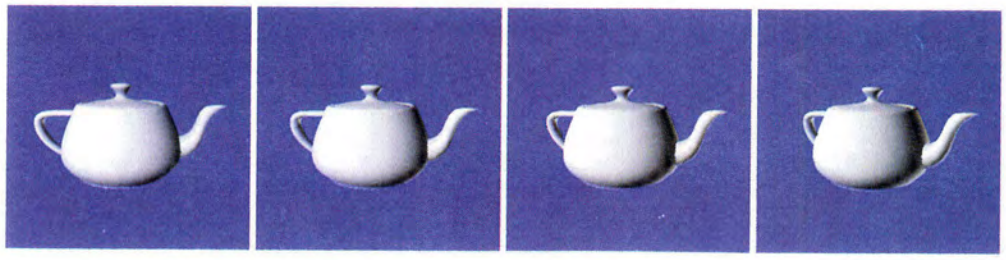


Fig.14 Teapot images generated by interpolation (two middle ones).

[2]Figure 11 and 13 images were digitally enlarged 200% with bicubic interpolation.

Spatial Anti-aliasing for Animation Sequences with Spatio-temporal Filtering

Mikio Shinya
NTT Human Interface Laboratories
3-9-11 Midori-cho, Musashino-shi
Tokyo 180, Japan
email: shinya@nttarm.ntt.jp
tel: +81 422 59 2648

Abstract

Anti-aliasing is generally an expensive process because it requires super-sampling or sophisticated rendering. This paper presents a new type of anti-aliasing filter for animation sequences, the *pixel-tracing filter*, that does not require any additional sample nor additional calculation in the rendering phase. The filter uses animation information to calculate correlation among the images, and sub-pixel information is extracted from the sequence based on the correlation. Theoretical studies prove that the filter becomes an ideal anti-aliasing filter when the filter size is infinite.

The algorithm is simple image processing implemented as post-filtering. The computational cost is independent of the complexity of the scene. Experiments demonstrate the efficiency of the filter. Almost complete anti-aliasing was achieved at the rate of about 30 seconds per frame for very complex scenes at a resolution of 256 × 256 pixels. The pixel tracing filter provides effective anti-aliasing for animation sequences at a very modest computational cost.

CR Categories and Subject Descriptors: I.3.3 [Computer Graphics]: Picture/Image Generation; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism

Additional Keywords and Phrases: Anti-aliasing, Spatio-temporal filtering, Computer Animation

1 Introduction

Aliasing artifacts have been troublesome in the field of graphics for a long time. These problems are particularly bad in animation sequences, since flickering thin objects and traveling jaggies are very noticeable.

For viewers in general, these spatio-temporal artifacts are more noticeable than the purely spatial ones in still images. To detect spatial aliasing, the *true* images (e.g., continuous lines or checker board patterns) should be inferred from the sampled image by intelligent, high-level visual processing. On the other hand, spatio-temporal aliasing can be detected by low-level vision processes (e.g., flicker detection and optical flow segmentation) without deep knowledge. This may seem rather negative, but it also implies a positive aspect: there may be easier ways to detect and remove aliasing in animation sequences.

Usually, there is strong correlation among the successive frames of motion pictures. This correlation allows efficient image compression in video codecs (coder/decoder) [NETRA]. This motivates us to extract sub-pixel information from image sequences, which could reduce aliasing artifacts.

This paper mathematically analyzes spatio-temporal characteristics of motion image sequences, and clarifies the useful features of their spectrum. Based on the analysis, a new type of anti-aliasing algorithm is proposed. In the algorithm, the image sequences are filtered with a linear shift-variant spatio-temporal filter called the *pixel-tracing filter*. Through the image sequence, the filtering process traces the pixels corresponding to the same object point, and the weighted sum of their colors is calculated. Theoretical studies prove that the filter acts as an ideal anti-aliasing filter when the filter size is infinite.

Unlike most anti-aliasing algorithms, this algorithm is achieved by post-filtering. The advantages are:

- fast execution independent of the scene complexity (e.g., number of polygons),
- simplicity of implementation,
- no dependence on the rendering process.

Experiments showed that the algorithm was efficient in terms of computational cost and provided effective image improvement.

2 Related Work

There are too many studies of anti-aliasing to review exhaustively, so only spatio-temporal approaches are briefly mentioned here. There are two major methods of spatio-temporal anti-aliasing: super-sampling and analytic calculation. In the super-sampling scheme, distributed ray tracing [COOK84] and alpha-blending [HAEBERLI] with stochastic sampling [DIPPE,COOK86] are the most successful and commonly used. Their advantages are simplicity and generality, but the disadvantage is a computational cost that is proportional to the rate of super-sampling. Although adaptive sampling [LEE] and optimal sampling patterns [MITCHELL] have been investigated, image improvement by super-sampling is generally computationally expensive. The analytic approach, on the other hand, is attractive because an exact solution can be calculated in relatively modest computation time. However, algorithms usually involve rather complicated processes, such as three-dimensional scan-conversion [GRANT] and analytic filtering of polygons [CATMULL84], and are only applicable to particular object primitives (typically polygons). In short, both approaches directly calculate sub-pixel or sub-frame information and then apply local filters.

Our approach differs from the above methods in three ways. First, our approach does not require any additional sample or additional calculation in the rendering phase. Second, it evaluates sub-pixel information from the image sequences themselves, taking the advantage of global spatio-temporal correlation. Third, our method uses a temporally global filter to removes spatial aliasing while other methods attempt to produce motion-blur by local temporal filtering.

3 Fourier Analysis

Temporal variation in animation sequences is usually due to the motion of the camera and objects. In this section, we mathematically analyze the spatio-temporal spectra of image sequences of moving objects. The velocity on the image plane is first assumed to be constant in time and space; analyses with spatial and temporal variation follow. The analyses provide an ideal anti-aliasing filter with infinite integral under certain conditions. Throughout this section, a one-dimensional space (image) is assumed for simplicity, but extension to two-dimensional images is mathematically straightforward.

3.1 Preparation

Let x be the image coordinate in pixels and t be the time in frames. Let a real function $f_0(x)$ be the image at $t = t_0$, and $f(x;t)$ be the image sequence. The spatial Fourier transform of f is defined by

$$\begin{aligned} F_0(\xi) &= \int f_0(x)\exp(\imath\xi x)dx, \\ F(\xi;t) &= \int f(x;t)\exp(\imath\xi x)dx, \end{aligned}$$

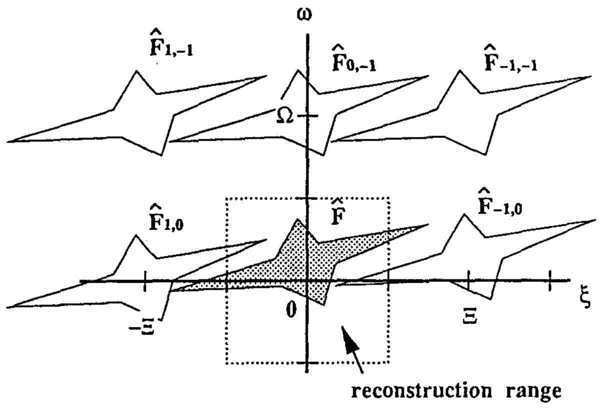


Figure 1: Aliasing in the Fourier Domain.

where ξ denotes the spatial angular frequency (rad/pixel), and $\imath$ is the imaginary unit, $\imath^2 = -1$. Similarly, the temporal Fourier transform is defined by

$$\hat{F}(\xi,\omega) = \int F(\xi;t)\exp(\imath\omega t)dt,$$

where ω is the temporal angular frequency in rad/frame.

The sampled image sequence $f_s(x,t)$ is represented by

$$f_s(x;t) = f(x;t)\sum_{k,l}\delta(x - 2\pi k/\Xi)\delta(t - 2\pi l/\Omega),$$

where Ξ and Ω are the sampling frequencies in space and time. When one point per pixel per frame is sampled, $\Xi = 2\pi$, and $\Omega = 2\pi$. The Fourier transform of f_s is

$$\hat{F}_s(\xi,\omega) = \sum_{n,m}\hat{F}_{n,m}(\xi,\omega), \tag{1}$$

where

$$\hat{F}_{n,m}(\xi,\omega) = \hat{F}(\xi + n\Xi, \omega + m\Omega).$$

Equation 1 indicates that replicas of $\hat{F}$ appear, centered at the grid points $(-n\Xi, -m\Omega)$, as illustrated in Figure 1.

When $\hat{F}(\xi,\omega) \neq 0$ outside the Nyquist frequencies $(\pm\Xi/2, \pm\Omega/2)$, some replicas intrude on the reconstruction range, causing aliasing artifacts. In other words, anti-aliasing can be achieved if replicas $\hat{F}_{n,m}$ can be filtered out. Therefore, anti-aliasing can be regarded as a process which calculates filtered images from the sampled images, and consequently, our objective is to find some mapping

$$f_s \longmapsto \int f_0(x)w(x_0 - x)dx$$

for any x_0. Here, $w(x)$ denotes some desirable spatial anti-aliasing filter.

The notation defined here is listed in Table 1. An introduction to sampling theory and aliasing can be found in [FOLEY].

Table 1: Symbols and notation

x	position on the image (pixel)
t	time (frame)
ξ	spatial angular frequency (rad/pixel)
ω	temporal angular frequency (rad/frame)
$f_0(x)$	image at $t = t_0$
$f(x;t)$	image at t
$f_s(x;t)$	sampled image sequence
$F_0(\xi)$	the spatial spectrum of f_0
$\hat{F}(\xi,\omega)$	the spatio-temporal spectrum of f.
$\hat{F}_s(\xi,\omega)$	the spatio-temporal spectrum of f_s.
Ξ	spatial sampling frequency
Ω	temporal sampling frequency
$\hat{F}_{n,m}$	the replica of $\hat{F}$ centered at $(-n\Xi, -m\Omega)$
$w(x)$	spatial anti-aliasing filter
$g(x,t)$	shift variant spatio-temporal filter
$\hat{G}(\xi,\omega)$	the spatio-temporal spectrum of w

3.2 Constant Velocity Motion

First, let us consider the simplest motion, constant velocity motion. In this case, the image at t can be represented by

$$f(x;t) = f_0(x + v_0(t_0 - t)), \qquad (2)$$

where v_0 is the velocity of the pattern. Its spatio-temporal spectrum is

$$\begin{aligned}\hat{F}(\xi,\omega) &= \int \exp(\imath\omega t)dt \int f_0(x + v_0(t_0 - t)) \exp(\imath\xi x)dx \\ &= \int F_0(\xi) \exp(\imath\xi v_0(t - t_0)) \exp(\imath\omega t))dt \\ &= 2\pi F_0(\xi) \exp(-\imath\xi v_0 t_0)\delta(v_0\xi + \omega), \qquad (3)\end{aligned}$$

where δ is Dirac's delta function and we used the equality

$$\int \exp(\imath uv)dv = 2\pi\delta(u).$$

Equation 3 clarifies a very important fact: *the spectrum $\hat{F}$ can be separated from the replicas even though the spatial spectrum $F_0(\xi)$ ranges beyond the Nyquist frequency.* Figure 2 illustrates this situation. The replicas can be filtered out as shown in the figure if velocity v_0 is known. Fortunately, the velocity can be easily calculated from animation data in graphics applications. Thus, an ideal anti-aliasing filter in this case looks like[1]

$$\hat{G}_v(\xi,\omega) = 2\pi\delta(v_0\xi + \omega). \qquad (4)$$

The linear filtering in the Fourier domain $\hat{G}_v\hat{F}_s$ is equivalent to convolution in real time-space, that is,

$$\int\int f_s(x,t)\delta((x_0 - x) - (t_0 - t)v_0)dxdt. \qquad (5)$$

This motivates us to study more general cases.

[1]Strictly speaking, the filter $\hat{G}_v$ involves a convergence problem because infinite animation sequences are assumed here. This will be solved in the next section.

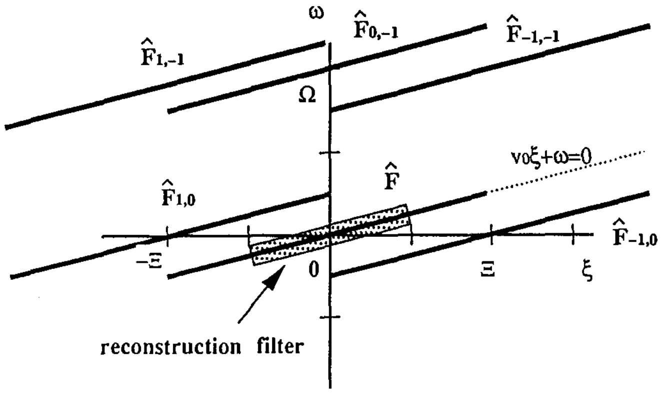


Figure 2: Spatio-temporal spectrum of constant velocity motion.

3.3 General Motion

Let us consider general motion. When the image point x_0 at t_0 moves to x_1 at t_1, we denote the motion by

$$x_1 = \chi(t_1; x_0, t_0). \qquad (6)$$

For example, the flow χ_v for constant motion is:

$$\chi_v(t; x_0, t_0) = x_0 + v(t - t_0).$$

Note that the reciprocity generally holds from the definition

$$x = \chi(t; \chi(t_0; x, t), t_0).$$

To avoid convergence problems, a finite animation sequence should be considered. With the flow χ, the sequence can be described as:

$$f(x,t) = \begin{cases} f_0(\chi(t_0; x, t)) & \text{if } t \in [-T/2, T/2] \\ 0 & \text{otherwise,}\end{cases}$$

where T is the length of the animation. The sampled image sequence are represented by

$$\begin{aligned} f_s(x,t) &= f(x,t)\sum_{k,l}\delta(x - 2\pi k/\Xi)\delta(t - 2\pi l/\Omega), \\ \hat{F}_s(\xi,\omega) &= \sum_{n,m}\int_{-T/2}^{T/2} dt \int_{-\infty}^{\infty} f_0(\chi(t_0; x, t)) \\ &\quad \exp(\imath(\xi + n\Xi)x + \imath(\omega + m\Omega)t)dx \\ &= \sum_{n,m}\hat{F}_{n,m}.\end{aligned}$$

Next, let us consider the anti-aliasing filter g. Filtering for constant motion, Eq. 5, can be rewritten as

$$\int\int f_s(x,t)\delta(x_0 - \chi_v(t_0; x, t))dxdt.$$

By analogy, we set our filter kernel g as

$$\begin{aligned} g(x,t) &= (1/T)w(x_0 - \chi(t_0; x, t))(\partial\chi/\partial x)_{t_0,t} \\ &= (1/T)w(x_0 - \chi(t_0; x, t))D_\chi(t_0; x, t) \qquad (7)\end{aligned}$$

for space-variant filtering at (x_0, t_0):

$$h(x_0,t_0) = \int\int f_s(x,t)g(x,t)dxdt. \qquad (8)$$

Here, $w(x)$ represents some appropriate anti-aliasing filter, such as a sinc-function, Gauss function, box function, and so on. The factor $1/T$ is the normalization constant, and $D_\chi = (\partial\chi/\partial x)$ compensates for image magnification variation due to spatially non-uniform motion.

Now, we prove that the filtering defined by Eq. 8 becomes an ideal anti-aliasing filter in the limit that $T \to \infty$. From the Parseval Identity, Eq. 8 can be rewritten as

$$\begin{aligned} h(x_0,t_0) &= (1/2\pi)^2 \int\int \hat{F}_s(\xi,\omega)\hat{G}^*(\xi,\omega)d\xi d\omega \\ &= (1/2\pi)^2 \sum_{n,m} \int\int \hat{F}_{n,m}(\xi,\omega)\hat{G}^*(\xi,\omega)d\xi d\omega \\ &= \sum_{n,m} h_{n,m}, \end{aligned}$$

where $\hat{G}^*$ denotes the complex conjugate of $\hat{G}$. The function $\hat{G}$ is the spatio-temporal spectrum of g, calculated by

$$\begin{aligned} \hat{G}(\xi,\omega) &= (1/T)\int\int w(x_0 - \chi(t_0;x,t))(\partial\chi/\partial x) \\ &\quad \exp(\imath(\xi x + \omega t))dtdx \\ &= (1/T)\int\int w(x_0 - u) \\ &\quad \exp(\imath\chi(t;u,t_0)\xi)\exp(\imath\omega t)dudt, \end{aligned}$$

where $u = \chi(t_0;x,t)$. Then, the integral $h_{n,m}$ can be evaluated as

$$\begin{aligned} h_{n,m} &= 1/(2\pi)^2(1/T)\int_{-T/2}^{T/2} \exp(\imath(\omega + m\Omega)t_1)dt_1 \\ &\quad \int f_0(\chi(t_0;x_1,t_1))\exp(\imath(\xi + n\Xi)x_1)dx_1 \\ &\quad \int\int w(x_0 - u)\exp(-\imath\chi(t_2;u,t_0)\xi) \\ &\quad \exp(-\imath\omega t_2)dudt_2 \\ &\quad \int d\xi \int d\omega \\ &= (1/T)\int_{-T/2}^{T/2} \exp(\imath m\Omega t_1)dt_1 \\ &\quad \int f_0(\chi(t_0;x_1,t_1))\exp(\imath n\Xi x_1)dx_1 \\ &\quad \int\int w(x_0 - u)\delta(t_1 - t_2)\delta(x_1 - \chi(t_2;u,t_0))dudt_2 \\ &= \int w(x_0 - u)f_0(u)du \\ &\quad \int_{-T/2}^{T/2} \exp(\imath n\Xi\chi(t_1;u,t_0)\exp(\imath m\Omega t_1)dt_1/T, \end{aligned}$$

where we used the reciprocity $\chi(t_0;\chi(t_1;u,t_0),t_1) = u$. Consequently,

$$\lim_{T\to\infty} h_{n,m} = \int w(x_0 - u)f_0(u)du(\lim_{T\to\infty} K_n(m\Omega;u)/T),$$

where $K_n(\omega;u)$ is the Fourier transform of the function k_n,

$$k_n(t;u) = \exp(\imath n\Xi\chi(t;u,t_0)).$$

Obviously,

$$h_{0,0} = \int w(x_0 - u)f_0(u)du.$$

On the other hand, when K_n is not singular at $\omega = m\Omega$, the aliasing pattern tends to 0, as

$$\lim_{T\to\infty} h_{n,m} = 0.$$

This completes the proof.

Note that $K_n(m\Omega, u)$ can be singular when, for example, motion is periodic with a frequency of $(m\Omega/n)$, or constant motion with a velocity of $(m\Omega/n\Xi)$.

3.4 Discrete Filtering

The filtering Eq. 8 can also be represented in a discrete form. By setting $\Xi = 2\pi$ and $\Omega = 2\pi$ (1 sample/pixel/frame sampling), we have

$$\begin{aligned} h(x_0,t_0) &= \int\int f_s(x;t)g(x,t)dxdt \\ &= (1/T)\int_{-T/2}^{T/2} dt \int_{-X/2}^{X/2} f(x;t)g(x,t) \\ &\quad \sum_{k,l} \delta(x-k)\delta(t-l)dtdx \\ &= (1/T)\sum_{k=-X/2}^{X/2}\sum_{l=-T/2}^{T/2} f(k;l)w(x_0 - \chi(t_0;k,l)) \\ &\quad D_\chi(t_0;k,l) \end{aligned} \qquad (9)$$

for T-frame image sequences at the X pixel image resolution. Since Eq. 9 is a finite weighted sum of the sampled images, it can be directly computed.

The magnification factor $D_\chi = (\partial\chi/\partial x)$ compensates for image distortion due to non-uniformity of motion flow. For spatially uniform motion (more generally, incompressible flow), $D_\chi \equiv 1$. Furthermore, since $D_\chi(t_0;t,x) \to 1$ as $t \to t_0$, we can assume

$$D_\chi \simeq 1,$$

when the filter size T is small. If non-uniformity is not negligible, we have to evaluate D_χ point by point. Analytic formulae for D_χ are given in the Appendix.

For practical implementation, we slightly modify the filtering equation Eq. 9. By assuming local uniformity of motion flows, we have

$$h(x_0,t_0) = (1/T)\sum_{k,l} f(k;l)w(\chi(l;x_0,t_0) - k), \qquad (10)$$

where we used the uniformity $x - y = \chi(t;x,t') - \chi(t;y,t')$. The advantage of Eq. 10 over Eq. 9 is that only one flow $\chi(l;x_0,t_0)$ should be traced for (x_0,t_0) rather than all flows $\chi(t_0;l,k)$.

The normalization factor $(1/T)$ relies on

$$\lim_{T\to\infty}(1/T)\sum_{l=-T/2}^{T/2}\sum_{k} w(\chi(l;x_0,t_0)-k)=1,$$

and would cause a normalization problem for finite T. Thus, it is better to adopt explicit normalization such as

$$h(x_0,t_0)=\sum_{k,l} f(k;l)w(\chi(l;x_0,t_0)-k)/\sum_{k,l} w(\chi(l;x_0,t_0)-k). \tag{11}$$

4 Algorithm

This section shows a simple algorithm for applying the anti-aliasing filter. When only one velocity field occupies the image, the only problem is to calculate the flow $\chi(t;x_0,t_0)$. However, when more than two velocity fields overlap, the filter should be separately applied because the theories rely on the uniqueness of the field. This happens when the projections of differently moving objects overlap (Figure 3).

Thus, the keys to the implementation are how to evaluate the motion flow χ and how to separate fields of different velocity. To deal with multiple flows, we adopt the filtering equation Eq. 11, assuming local uniformity of flows.

Data From animation models, we receive animation data for the sequence, such as transformation of objects and camera parameters. From the rendering process, RGB values, z-values, and object-id values are provided for each pixel at each frame, for example, in the form of G-buffers [SAITO]. Here, the object-id's are only used to identify object motion, and can be omitted for walk-through scenes. Let us denote these values for the pixel (k_x,k_y) at the frame l by rgb$[k_x][k_y][l]$,z$[k_x][k_y][l]$, and id$[k_x][k_y][l]$, respectively.

As the work space to capture multiple flows, we have a list structure of rgb, z, and α for each pixel, denoted by rgb_{flow}[ix][iy], $z_{flow}[ix][iy]$, and $\alpha_{flow}[ix][iy]$.

Pixel Tracing We now treat two-dimensional images. Let us denote two-dimensional vectors and their x,y-components by using the arrow and suffix notation, such as $\vec{k}=(k_x,k_y)$.

The motion flow, $\vec{\chi}(l;\vec{k}_0,l_0)=(\chi_x,\chi_y)$, corresponding to the sample point $\vec{k}_0=(k_{0x},k_{0y})$ at $t=l_0$, can be easily calculated from the animation information, the object-id, A, and the z-value (Figure 3). Let the transformation from the object coordinate of Object A to the screen space at t be $T_A(t)$. Then, the corresponding object point $p_A=(x_A,y_A,z_A,w_A)$ is given by

$$p_A=(k_{0x},k_{0y},z[k_{0x}][k_{0y}][l_0],1)T_A^{-1}(l_0).$$

At $t=l$, the object point p_A is projected by $T_A(l)$, and thus, the flow $\vec{\chi}$ and the corresponding depth $\zeta(l;\vec{k}_0,l_0)$ can be calculated as

$$\begin{aligned}\vec{\chi}(l;\vec{k}_0,l_0) &= (x/w,y/w)\\ \zeta(l;\vec{k}_0,l_0) &= z/w.\\ (x,y,z,w) &= p_A T_A(l_0)\\ &= (k_{0x},k_{0y},z[k_{0x}][k_{0y}][l_0],1)\\ &\quad T_A^{-1}(l_0)T_A(l)\end{aligned} \tag{12}$$

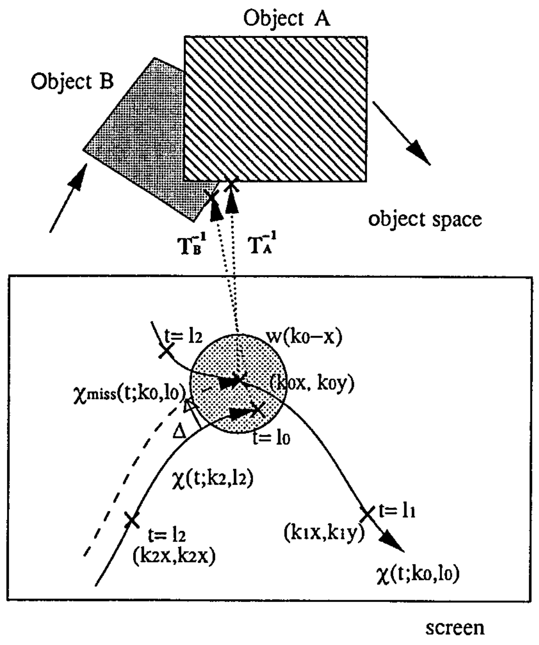


Figure 3: Pixel-tracing.

When the sample point misses an object, the filtering can be applied in the following way. In the example in the Figure 3, Object B fails to hit the sample point at $\vec{k}_0, t=l_0$, but pixel tracing from $\vec{k}_2=(k_{2x},k_{2y}), t=l_2$, reveals that Object B should exist in the reconstruction area of $\vec{k}_0, t=l_0$ because the traced point $\chi(l_0;\vec{k}_2,l_2)$ lies in the area. Therefore, we trace the flow for $\vec{k}_0$ by

$$\vec{\chi}_{miss}(l;\vec{k}_0,l_0)=\vec{\chi}(l;\vec{k}_2,l_2)+\Delta, \tag{13}$$

where

$$\Delta=\vec{k}_0-\vec{\chi}(l_0;\vec{k}_2,l_2).$$

Separation and Summation To separate different velocity fields, we adopt a simple rule, that is, *when the difference between two flows is smaller than some threshold, we regard them as the same flow.*

This can be described as follows. If both of the inequalities,

$$\|\vec{k}_0-\vec{\chi}(l_0;\vec{k}_1,l_1)\|<d_{th}, \tag{14}$$

and

$$\|\vec{k}_1-\vec{\chi}(l_1;\vec{k}_0,l_0)\|<d_{th}, \tag{15}$$

hold, the two sampled data are judged to belong to the same flow. Here, $\|\cdot\|$ denotes some norm on the image plane. The threshold d_{th} can be determined, for example, according to the diameter of the support of the filter kernel $w(\vec{x})$.

In the example in Figure 3, Inequality 15 is not true, so the two samples are processed as different flows. Note that the projection points of the same object do not necessarily belong to the same flow because of perspective.

With this criteria function, same_flow(), which returns 1 when two samples are judged as being in the same flow and 0 otherwise, the actual filtering for each velocity flow becomes

$$\begin{aligned}
\mathrm{rgb}_{flow} &= \sum_{l}\sum_{\vec{k}} \mathrm{same\_flow}(\vec{k}, l; \vec{k}_0, l_0) w(\vec{\chi}(l; \vec{k}_0, l_0) - \vec{k}) \\
&\quad \mathrm{rgb}[k_x][k_y][l] / \sum \mathrm{same\_flow}() w() \\
\alpha_{flow} &= \sum \mathrm{same\_flow}() w() / \sum w(), \\
\mathrm{z}_{flow} &= \min(\mathrm{z}_{flow}, \zeta(l; \vec{k}_0, l_0)) \qquad (16)
\end{aligned}$$

Here, α represents the 'coverage' of this flow. Note that the summation with $\vec{k}$ can be calculated only in a neighborhood of the flow $\vec{\chi}$ when the filter $w(\vec{x})$ is compactly supported.

At each pixel, we store the calculated RGB values, α-values, and z-values for all flows as a list, like the A-buffer structure [CARPENTER]. After the filtering, we sort the list with respect to the z-values at each pixel, and the final RGB values are determined by simple α-blending in the order of the sorted list, from near to far,

$$\mathrm{rgb}_{final} = \alpha_{flow1}\mathrm{rgb}_{flow1} + (1 - \alpha_{flow1})\alpha_{flow2}\mathrm{rgb}_{flow2} + \ldots. \qquad (17)$$

Procedure The procedure for filtering the frame l_0 can be summarized in the following way.

1) For each sample, $\vec{k}_0$, at frame $l = l_0$, do the following for all frames l within the filter.
   i) Calculate $\vec{\chi}(l; \vec{k}_0, l_0)$ according to Eq. 12.
   ii) Calculate the weighted sum according to Eq. 16.
2) For $l \neq l_0$ within the filter, do the following for all the samples, $\vec{k}$.
   i) Calculate the flow $\vec{\chi}(l_0; \vec{k}, l)$ according to Eq. 12.
   ii) For all samples $\vec{k_0}'$ at l_0 such that $w(\vec{k}_0' - \vec{\chi}(l_0; \vec{k}, l)) \neq 0$, do the following.
      a) If $\vec{k}, l$ belongs to any flow listed for $\vec{k}_0', l_0$, skip b) and c).
      b) Calculate the flow $\vec{\chi}_{miss}(l'; \vec{k}_0', l_0)$ according to Eq. 13, and calculate the weighted sum according to Eq. 16.
      c) Append the result to the list of $\vec{k}_0', l_0$.
3) For each pixel at l_0, sort the resulting list with respect to the z-values and apply alpha-blending according to Eq. 17.

As this filter involves a pixel tracing process, we call it the *pixel-tracing filter*.

5 Experiments and Discussion

The first experiment shows the influence of filter size, T. The test pattern is an almost horizontal (1 degree from the horizon) thin rectangle with the width of 1/8 pixel, constantly moving in the vertical direction at a speed of $0.22 = 11/50$ pixel/frame. Note that $\lim_{T\to\infty}(h_{50,11}/T) \neq 0$.

Figures 4-a, -b, and -c show the original image, and the results of applying filters of various sizes. As the filter kernel $w(x)$, we used the box function with one pixel area. Figures 4-d and -e show the analytic solution and the root-mean-square error of the filtered results with respect to the filter size. As shown in the figure, effective anti-aliasing was achieved with a filter size 128. The remaining error comes from the replica $\hat{F}_{50,11}$.

The second experiment shows an example of non-uniformly accelerated motion. The test pattern is rotating radial thin rectangles. As shown in Figure 5, aliasing was mostly removed with the 32-frame filter.

The final experiment shows application to a more practical image sequence taken from a walk-through scene in 'Také Tera.' In the sequence, only the camera moves and everything else is fixed in space. The original images were synthesized by using the GL library on the IRIS workstations at a resolution of 256 × 256. The scene consists of about 4M polygons. Figure 6 demonstrates the efficiency of the algorithm, where the severe aliasing artifacts seen in the original image were largely removed. The filter size was 16 frames.

The execution time is about 30 CPU seconds per frame on an IRIS Crimson R4000-50 at 256 × 256 resolution. The computation cost is directly proportional to $n_x \times n_y \times n_t$, where $n_x \times n_y$ is the image resolution, and n_t is the filter size. Considering that frame-by-frame recording onto a VCR takes about thirty seconds per frame, this powerful anti-aliasing is almost free! Furthermore, as the filtering is simple image processing with pixel-level independence, it might be possible to design parallel hardware to execute it in real-time, which would be attractive for visual simulators and virtual reality applications.

Future work includes application to reflected/refracted images, and coupling with stochastic sampling techniques. The algorithm relies on transformation between the screen space and the object space. Although conventional ray tracers cannot provide the transformation, the beam tracing/pencil tracing approach [HECKBERT,SHINYA] can calculate it in the form of system matrices and thus may be applicable to the filtering.

Since stochastic sampling techniques are powerful tools for anti-aliasing, it is an attractive idea to combine the two approaches. If we jitter the sample point of each pixel at each frame, the pixel-tracing filter acts exactly as a purely spatial filter for objects that are steady on the image plane. This means that spatial stochastic super-sampling can be performed by the pixel-tracing filter with only one point per pixel per frame sampling. This could also reduce the problems with constant velocity motion in the case of $n\Xi v_0 = m\Omega$, which we observed in Figure 4.

6 Conclusion

A new type of efficient anti-aliasing filter, the pixel-tracing filter, was proposed for animation sequences. The filter sums sub-pixel information using the correlation among images calculated from animation information. Theoretical studies prove the ability of the filter, and experimental results demonstrate the efficiency.

The algorithm is simple image processing implemented as post-filtering. The computational complexity is of constant order with regard to the complexity of scenes (e.g., number of polygons). With the pixel-tracing filter, effective anti-aliasing can be completed for animation sequences with a very modest computational cost.

Acknowledgments

The author would like to thank the Siggraph reviewers, whose comments greatly contributed to improving the theoretical part. He also wishes to thank Takahiko Kamae, Rikuo Takano, and Kazuyoshi Tateishi for their administrative support and encouragement, Atsushi Kajiyama for his technical support, and Toki Takahashi, Taka Saito, and Toshi Tanaka for helpful discussion.

References

[CARPENTER] Loren Carpenter, 'The A-buffer, An Antialiased Hidden Surface Method,' Computer Graphics **18**, No.3, pp.103-108, 1984.

[CATMULL84] Edwin Catmull, 'An Analytic Visible Surface Algorithm for Independent Pixel Processing,' Computer Graphics **18**, No.3, pp.109-115, 1984.

[COOK84] R. L. Cook, T. Porter, L. Carpenter, 'Distributed Ray Tracing,' Computer Graphics **18**, No.3, pp.137-145, 1984.

[COOK86] R. L. Cook, 'Stochastic Sampling in Computer Graphics,' ACM Trans. Graphics, 5, No.1, pp.51-57, 1986.

[DIPPE] M. A. Dippé, 'Anti-aliasing through Stochastic Sampling,' Computer Graphics **19**, No.3, pp.69-78, 1985.

[FOLEY] James D. Foley, Andies van Dam, Steven K. Feiner, John F. Hughes, 'Computer Graphics Principal and Practice,' Addison-Wesley, 1990.

[GRANT] Charles W. Grant, 'Integrated Analytic Spatial and Temporal Anti-Aliasing for Polyhedra in 4-Space,' Computer Graphics **19**, No.3, pp.79-84, 1985.

[HAEBERLI] P. Haeberli, K. Akeley, 'The Accumulation Buffer: Hardware Support for High-Quality Rendering,' Computer Graphics, **24**, No.4, pp.309-318, 1990.

[HECKBERT] P. S. Heckbert, P. Hanrahan, 'Beam Tracing Polygonal Objects,' Computer Graphics, 18, No.3, pp.119-128, 1984.

[LEE] Mark E. Lee, Richard A. Redner, and Samuel P. Uselton, 'Statistically Optimized Sampling for Distributed Ray Tracing,' Computer Graphics **19**, No.3, pp.61-67, 1985.

[MITCHELL] D. Mitchell, 'Spectrally Optimal Sampling for Distributed Ray Tracing,' Computer Graphics **25**, No.4, pp.157-164, 1991.

[NETRA] A. N. Netravali and B. G. Haskell, 'Digital Pictures - Representation and Compression,' Prenum Press, 1988.

[SAITO] Takafumi Saito and Toki Takahashi, 'Comprehensible Rendering of 3-D Shapes,' Computer Graphics **24**, No.4, pp.197-206, 1990.

[SHINYA] M. Shinya, T. Takahashi, and S. Naito, 'Principles and Applications of Pencil Tracing,' Computer Graphics, **21**, No.4, pp. 45-54, 1987.

Appendix: Calculation of D_χ

Here, we derive D_χ in Eq. 9 for two-dimensional images. We assume motion flows $\vec{\chi}(t; x_0, y_0, t_0)$ represented by Eq. 12. In the two-dimensional case, D_χ becomes the Jacobian of $\vec{\chi}$,

$$\begin{aligned} D_\chi &= \begin{vmatrix} (\partial\chi_x/\partial x_0)_{y0} & (\partial\chi_x/\partial y_0)_{x0} \\ (\partial\chi_y/\partial x_0)_{y0} & (\partial\chi_y/\partial y_0)_{x0} \end{vmatrix} \\ &= (\partial\chi_x/\partial x_0)_{y0}(\partial\chi_y/\partial y_0)_{x0} \\ &\quad -(\partial\chi_x/\partial y_0)_{x0}(\partial\chi_y/\partial x_0)_{y0}. \end{aligned}$$

By setting the translation matrix

$$T_A^{-1}(t_0)T_A(t) = \{\tau_{ij}\},$$

the partial deviation $(\partial\chi_x/\partial x_0)_{y0}$, etc., can be calculated as

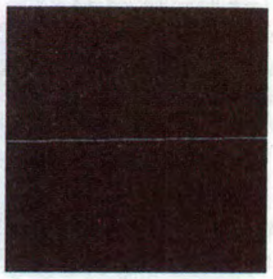
$$\begin{aligned} (\partial\chi_x/\partial x_0)_{y0} &= (1/w)(\partial x/\partial x_0)_{y0} - (x/w^2)(\partial w/\partial x_0)_{y0} \\ &= (1/w)(\tau_{11} - (n_x/n_z)\tau_{31}) - (x/w^2) \\ &\quad (\tau_{14} - (n_x/n_z)\tau_{34}), \end{aligned}$$

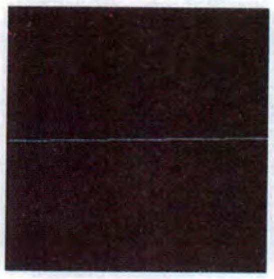
where $\vec{n} = (n_x, n_y, n_z)$ is the normal vector of the object surface at p_A, and we used

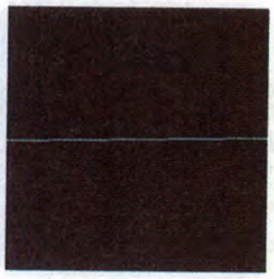
$$\begin{aligned} (\partial x/\partial x_0)_{y0} &= (\partial x/\partial x_0)_{y0,z0} + (\partial z_0/\partial x_0)_{y0}(\partial x/\partial z_0)_{x0,y0} \\ &= \tau_{11} - (n_x/n_z)\tau_{31}, \end{aligned}$$

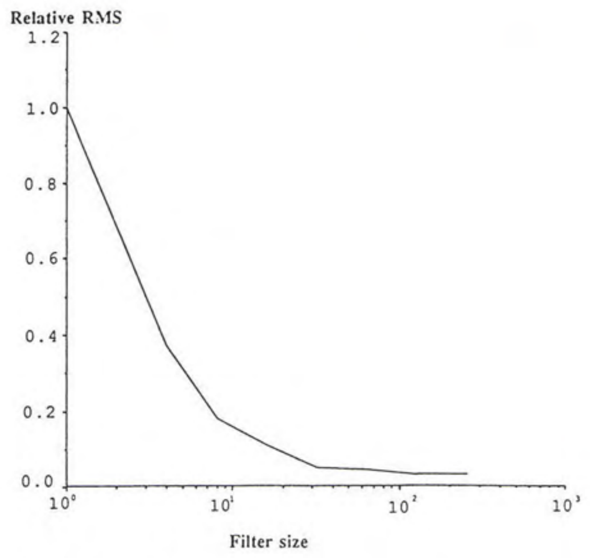
and so on.

(a) Original


(b) Filter size = 32

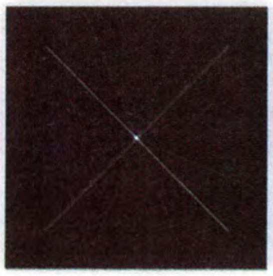

(c) Filter size = 64


(d) Ananlytic solution



(e) Relative root-mean-square error

Figure 4: Thin rectangle.


(a) Original image

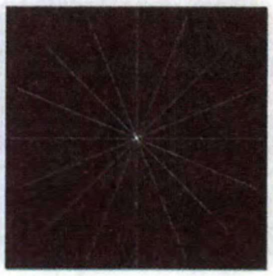
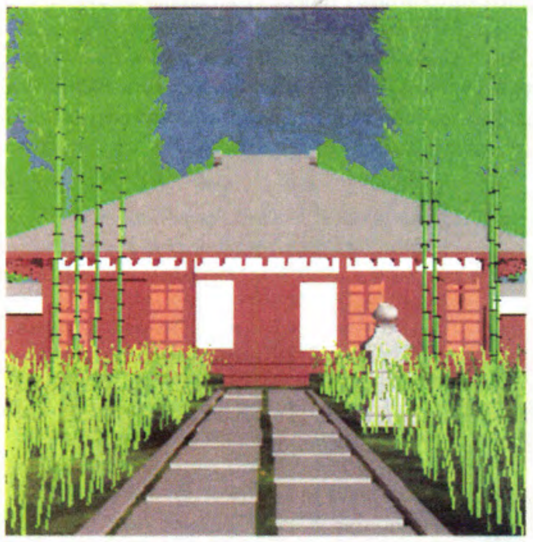

(b) Filtered image

Figure 5: Rotating thin rectangles.


(a) Original image

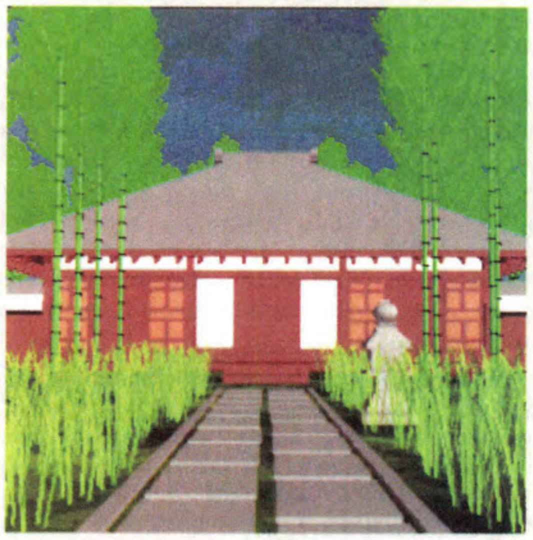

(b) Filtered image

Figure 6: Také Tera(Bamboo Temple).

Motion Compensated Compression of Computer Animation Frames *

Brian K. Guenter †, Hee Cheol Yun, and Russell M. Mersereau ‡

Abstract

This paper presents a new lossless compression algorithm for computer animation image sequences. The algorithm uses transformation information available in the animation script and floating point depth and object number information stored at each pixel to perform highly accurate motion prediction with very low computation. The geometric data, i.e., the depth and object number, is very efficiently compressed using motion prediction and a new technique called direction coding, typically to 1 to 2 bits per pixel. The geometric data is also useful in z-buffer image compositing and this new compression algorithm offers a very low storage overhead method for saving the information needed for z-buffer image compositing. The overall compression ratio of the new algorithm, including the geometric data overhead, is compared to conventional spatial linear prediction compression and is shown to be consistently better, by a factor of 1.4 or more, even with large frame-to-frame motion.

CR Categories: I.4.2[compression(coding)]exact coding.
Additional keywords: compression,computer animation,computer graphics, motion prediction

1 Introduction

With the increasing popularity and falling cost of computer animation comes a new problem: storing the enormous data files which even short computer animation sequences require. Five minutes of NTSC resolution computer animation takes up approximately 8.5 gigabytes of storage; film resolution takes many times more. This amount of data cannot be economically stored on line in high speed secondary storage devices. Animation image files are typically stored off-line on removable media.

An alternative to using off-line storage is to compress the image data and store it on-line. This is very desirable for sequence editing and image manipulation, for example. For high image quality only lossless compression is acceptable; images can then be exactly reconstructed from their compressed representation. Errors do not accumulate if images are combined or manipulated and run through the compression decompression cycle several times.

Much more information is available to a compression algorithm for computer animation than is the case for live action video. However, surprisingly little work has been done on exploiting the information in a computer animation script to improve image compression efficiency. Previous work such as that described in [1] and [5] is actually image based compression although the application is to computer animation.

The lossless compression algorithm for computer animation to be described in this paper combines elements of both motion prediction and spatial linear prediction compression techniques, using each when most appropriate. The new compression algorithm uses transformation information in the animation script to perform essentially perfect image space motion prediction with very low computation. This is a major advantage of the new algorithm because motion prediction with subpixel accuracy based only the information present in the image sequence is computationally expensive [6][3]. Poor quality motion prediction increases the motion prediction error which reduces the maximum achievable compression ratio.

One of the most effective lossless image compression techniques is DPCM followed by entropy coding [4][8]. For typical live action video sequences the best compression achievable using this method is usually less than 2 to 1 [7]. The best computer generated images are nearly indistinguishable from real images so we can expect that good synthetic images will not compress any better than live action video.

For scenes with fairly rapid camera and object motion the compression ratio we have achieved with our new motion prediction compression is approximately 1.5 times that of spatial linear prediction compression techniques - about 3 to 1 compression with the new technique as opposed to 2 to 1 compression with DPCM. As camera and object motion decrease the compression ratio of the new technique steadily increases while spatial prediction compression remains constant at roughly 2 to 1.

Extra geometric information, the object number and the depth at each pixel, is stored in each frame to perform motion prediction. The geometric information is compressed very efficiently in our new algorithm, typically to 1 or 2 bits per pixel. For z-buffer compositing applications [2] this is another advantage of the new algorithm, because the depth information needed for z-buffer compositing is stored in very little space.

We assume the animation script contains a homogeneous matrix transformation for every object in every frame. The matrix transforms the object from the model space coordinate frame into the screen space coordinate frame. The transformation matrices are stored in an auxiliary file along with the compressed image data and constitute part of the overhead of the new compression algorithm. This limits the current implementation to rigid body motion but this is not an intrinsic limitation of the algorithm. Non-rigid body motion can be accommodated by storing appropriate transformation information, such as free form deformation

*This work was supported by the National Science Foundation under grant MIP-9205853

†Computer Animation Laboratory, GVU center, College of Computing, Georgia Institute of Technology, Atlanta GA 30332, E-mail: brian.guenter@cc.gatech.edu

‡Digital Signal Processing Labratory, School of Electrical Engineering, Georgia Institute of Technology, Atlanta GA 30332, E-mail: yun@eedsp.gatech.edu, rmm@eedsp.gatech.edu

mesh points for example [9], in the animation script.

The current implementation assumes that objects are represented as polygonal surfaces. Algorithms exist for converting many different surface representations to approximating polygonal surfaces. Many commercial image synthesis programs perform this conversion internally so the limitation to a polygonal representation is not unduly restrictive.

The geometric data has special properties we exploit to improve compression. As a consequence the coder is split into two parts: a geometrical data coder and a color data coder. General notation used throughout the paper is presented in Section 2.

Section 3 of the paper presents block diagrams of the algorithm. Section 4 describes the geometrical data coding algorithm. Section 5 describes the color data coding algorithm. Animation test results are presented in section 6 and conclusions and suggestions for further research are presented in section 7.

2 Notations and Data structure

In this paper the frame number, which is used to identify the specific frame, is expressed as a superscript. A subscript represents the object number when it is expressed as a single value and the spatial location when it is expressed as a pair of values. If the subscripts are omitted, that symbol represents the whole set of the corresponding data for that frame.

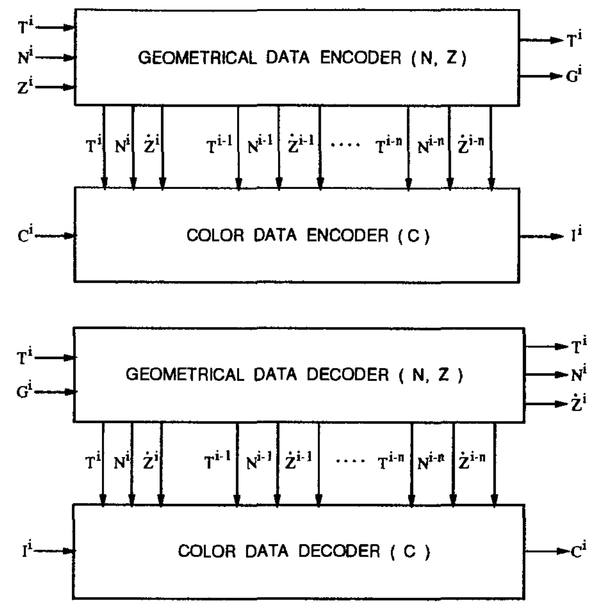
The data structure of a frame is divided into two parts. The first part is the set of 4×4 homogeneous matrices for all the objects $\{ T^i_j, j = 0..N_o - 1 \}$ by which the point in the model object space is transformed to the screen space. The other part is the 2-dimensional array of the data $P^i_{m,n}, m = 0..N_x - 1, n = 0..N_y - 1$, where N_o represents the number of objects and N_x and N_y represent the number of pixels in each direction. Each pixel datum $P^i_{m,n}$ is composed of the object number $N^i_{m,n}$, depth $Z^i_{m,n}$ and colors $C^i_{m,n}$ of the pixel at the spatial location (m, n). For example, the point in the i-th frame $(m, n, Z^i_{m,n})^T$ is transformed to the point $(x^j, y^j, z^j)^T$ in the *j-th* frame as follows:

$$T^j_k (T^i_k)^{-1} \begin{pmatrix} m \\ n \\ Z^i_{m,n} \\ 1 \end{pmatrix} = \alpha \begin{pmatrix} x^j \\ y^j \\ z^j \\ 1 \end{pmatrix} \qquad (1)$$

where $k = N^i_{m,n}$.

As mentioned above, the symbols without the subscripts represent the whole set of data for the frame. For example T^i stands for the set of matrices and N^i, Z^i, C^i represent the whole two dimensional array, also called a field, containing the object number, depth and color values of the *i-th* frame, respectively. The object number N^i and the depth Z^i are collectively called the geometrical data field. The color field C^i represents the R,G,B color fields, but sometimes can be used for one specific color field.

The object number and color values are represented as integers, but the depth is a real number. In our implementation, each of the RGB color values is usually represented by 8 bits/pixel (256 levels), and the depth is double precision floating point. Since the compression efficiency of the geometrical data is highly dependent on the accuracy of the calculation, the double precision representation is preferred. The required number of bits for the object number depends on the total number of objects.



GEOMETRICAL DATA ENCODER (N, Z)
COLOR DATA ENCODER (C)
GEOMETRICAL DATA DECODER (N, Z)
COLOR DATA DECODER (C)

Figure 1: *n-th* Order Frame Encoder and Decoder

3 System Block diagrams

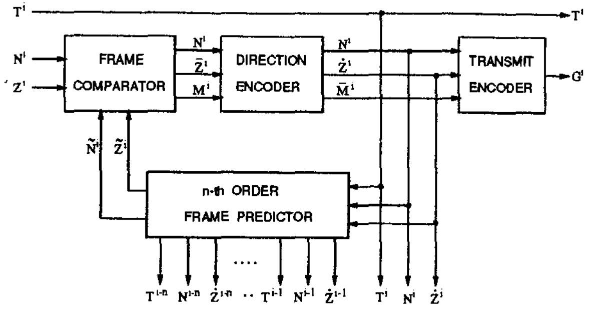
The heart of the coding scheme uses a linear predictive coding algorithm (DPCM) [4]. Since there exists substantial correlation between successive frames in computer animation as well as in real-life video, good compression gain can be achieved by these predictive schemes. Since both the object number and depth fields are needed to compute the motion trajectory of each pixel and these are encoded together into one data stream G^i, the whole system is divided into a geometrical data coding block for N^i, Z^i and a color data coding block for C^i as shown in Figs 1.

The object number N^i and color data C^i are coded losslessly, but the depth Z^i is allowed to contain error within a specified limit to achieve a high compression gain, because Z^i requires a relatively larger number of bits (64 bits/pixel for a double precision representation) than N^i, C^i.

The DPCM system requires storage for several frames determined by the order of the predictor. The geometrical data coding block stores the object number fields N^i and the depth fields $\dot{Z}^i$ of previous frames. Since only the decoded values are available for the depth field in the decoder, the geometrical data encoder uses the decoded depth field $\dot{Z}^i$ instead of the original depth field Z^i for correct reconstruction from the encoded data. The stored geometrical data $N^i, \dot{Z}^i$ are provided to the color data coding block to predict the color data in other frames by motion prediction.

4 Geometrical Data Coding

Figs 2 show the block diagrams of the encoder and decoder for geometrical data. The principle behind the geometrical data coding is that the geometrical data of the current frame is predicted from several previous frames which are compared to the current original frame pixel-by-pixel. Each pixel $P^i_{m,n}$ is classified as matched if $N^i_{m,n}$, $Z^i_{m,n}$ of the current frame are the same as $\tilde{N}^i_{m,n}$, $\tilde{Z}^i_{m,n}$ of the predicted frame, and unmatched otherwise. Since the object number and depth for the matched pixels can be recovered from the



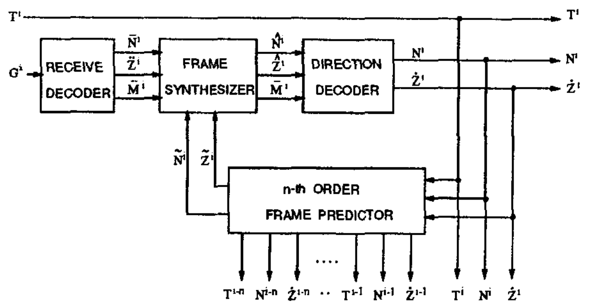


Figure 2: Geometrical Data Encoder and Decoder

predicted frame in the decoder, the only information that needs to be transmitted are the matching status field M^i which records whether or not each pixel is matched, and the complete geometrical data for the unmatched pixels. Unmatched pixels occur mainly in recently uncovered regions which cannot be predicted from previous frames, or from highly curved regions that are difficult to predict.

For the unmatched regions the geometrical data can be coded effectively by exploiting the spatial correlations between pixels, because pixels which belong to the same planar polygon satisfy the same plane equation. An algorithm called direction coding is proposed and described later. With direction coding, the unmatched pixels are classified into direction matched pixels and totally unmatched pixels. This matching information replaces $M^i_{m,n}$ at the unmatched pixel and this modified matching status field is $\bar{M}^i$. After the direction coding, since the majority of the frame is matched, the entropy of $\bar{M}^i$ is very small. Thus $\bar{M}^i$ can be compressed effectively by entropy coding or run-length coding and the original geometrical data $N^i_{m,n}$, $Z^i_{m,n}$ are transmitted in uncompressed form only for the totally unmatched pixels.

At the receive decoder, the matching status field $\bar{M}^i_{m,n}$, and the geometrical data $N^i_{m,n}$,$Z^i_{m,n}$ for totally unmatched pixels are obtained. For the matched pixels which can be identified by the $\bar{M}^i$, the object number $N^i_{m,n}$ and the depth $\tilde{Z}^i_{m,n}$ are copied from the predicted frame. Then for the unmatched pixels the $N^i_{m,n}$, $\acute{Z}^i_{m,n}$ are recovered by the direction decoder and the complete recovered frame data is fed into the frame predictor for the next frame prediction.

4.1 Frame Predictor

The set of four pixels $P^i_{m,n}$, $P^i_{m,n+1}$, $P^i_{m+1,n+1}$, $P^i_{m+1,n}$ is defined as a pixel square $S^i_{m,n}$. Each pixel square can be classified into one of three categories. The first is the plane pixel square where all four corner pixels come from the same planar polygon and make a planar square. The second is the adjacent polygon pixel square where four pixels are from the two adjacent polygons that share an edge that intersects the pixel square. The third is a non-adjacent polygon pixel square where the polygon boundary is across the pixel square but the polygons do not share an edge. These three cases are illustrated in Fig. 3.

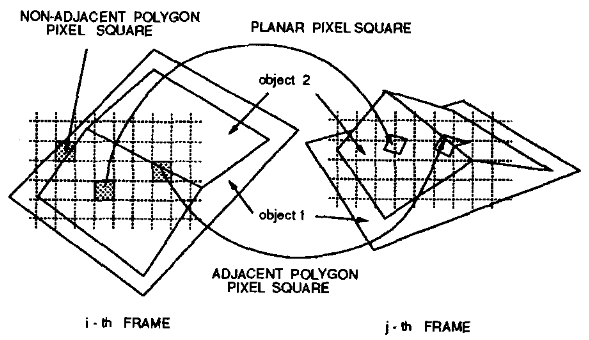

NON-ADJACENT POLYGON PIXEL SQUARE
PLANAR PIXEL SQUARE
object 2
object 1
ADJACENT POLYGON PIXEL SQUARE
i-th FRAME
j-th FRAME

Figure 3: Transform of Pixel Squares between Frames

The planar pixel square can be easily transformed by Eq. 1 and rendered into other frames. For the adjacent polygon pixel square, if the plane equations of the two polygons can be obtained by exploiting the neighboring pixel squares, then the pixel square can be partitioned into two polygons and each polygon can be transformed and rendered onto other frames in the same way. There is not enough information to make correct partitions for the non-adjacent polygon pixel squares, then these cannot be used to predict other frames. The pixel square $S^i_{m,n}$ can be defined as a planar pixel square if the four pixels satisfy the following plane conditions.

$$N^i_{m,n} = N^i_{m,n+1} = N^i_{m+1,n+1} = N^i_{m+1,n} \qquad (2)$$

$$|Z^i_{m,n} + Z^i_{m+1,n+1} - Z^i_{m,n+1} - Z^i_{m+1,n}| < \epsilon \qquad (3)$$

In Eq. 3 the inequality is used to deal with the error due to the limited precision of computation and the small number ϵ is determined as the allowable error for the depth of the pixels which are from the same planar polygon.

In some cases, the above two conditions are not enough to determine whether the pixel square is planar or not. For example, the pixels lying across the boundary of two separate polygons which are parallel to each other might satisfy these two conditions. There are several ways of reducing the possibility of an incorrect classification of a planar pixel square. One way is to add the following conditions, which test the relations between the depths of the surrounding pixels. If the following plane conditions are satisfied with regard to at least one corner of the pixel square, $S^i_{m,n}$ can be considered to be a plane pixel square.

$$|2Z^i_{l,k} - Z^i_{l-1,k} - Z^i_{l+1,k}| < \epsilon$$
$$|2Z^i_{l,k} - Z^i_{l,k-1} - Z^i_{l,k+1}| < \epsilon \qquad (4)$$
$$\text{for } l = m, m+1 \text{ and } k = n, n+1.$$

For the non-planar pixel square which does not satisfy the above plane conditions, if some two pixel squares around it are planar and if the intersection of those two planes are found to be across the pixel square by solving the plane equations of those two planes, then this pixel square is an adjacent polygon pixel square that can be divided into two polygons and transformed into other frames. One way to find the two plane pixel squares is to test the above plane conditions for each pair of pixel squares which are on the opposite sides of the current pixel square.

The geometrical data for most of a frame can be computed by transforming all the planar and adjacent polygon pixel squares of the previous frame into the current frame and rendering the transformed polygons on the frame buffer of geometrical data using the z-buffer algorithm.

Since the current frame does not change much from the previous frame, the transformed polygon of one pixel square is small and covers only a few pixels. Under this assumption, there are several effective techniques for geometrical data rendering. One simple method is to find the bounding box of the transformed pixel square and test whether each pixel point inside the bounding box is inside the polygon or not. For each inside pixel point, the depth of that point can be computed from the plane equation of the transformed polygon for the z-buffer rendering process. The back-face removal step might be applied before rendering.

There is one special case where the viewpoint and object are not moving. In this case the transform matrices of the current and previous frames are the same and the whole transform matrix $T_k^j(T_k^i)^{-1}$ will be the identity matrix in Eq. 1. For these pixels, the above complicated steps are not necessary and the only thing to do is simply to apply the depth of the previous frame to the z-buffer algorithm at the same pixel location. All these pixels are classified as matched pixels.

Due to occlusion some parts of the current frame may not be predictable from the previous frame. The percentage of predictable pixels can be increased by using higher order prediction. In the n-th order case, the previous n frames are transformed and rendered on the same frame buffers of object number and depth.

The frame predictor used in both the encoder and decoder have identical frame buffers for a object number and depth for higher order prediction. The encoder and decoder should store the same frame data in both predictors so that the predicted frames in both blocks will be the same.

4.2 Frame Comparator

The frame comparator compares the input frame data to the predicted frame data on a pixel-by-pixel basis and records the result in the matching status field M^i. If a pixel $P^i_{m,n}$ of the current frame and $\tilde{P}^i_{m,n}$ of the predicted frame satisfy the following conditions, then the pixel is considered to be predictable from the previous frames and said to be a matched pixel.

$$N^i_{m,n} = \tilde{N}^i_{m,n} \tag{5}$$

$$|Z^i_{m,n} - \tilde{Z}^i_{m,n}| < \epsilon \tag{6}$$

In Eq. 6, an inequality is used for the same reason as in Eq. 3. If a pixel $P^i_{m,n}$ is found to be matched, $M^i_{m,n}$ is set to 1 and otherwise set to 0. Because of the similarity between the adjacent frames, most of M^i will be 1. For matched pixels, the depth $Z^i_{m,n}$ of the input frame is replaced by the predicted depth $\tilde{Z}^i_{m,n}$ to guarantee the consistency of data between the encoder and decoder, because only $\tilde{Z}^i_{m,n}$ will be available in the decoder. By Eq. 6 the accuracy of the new depth $\tilde{Z}^i_{m,n}$ is guaranteed to be within ϵ. This modified depth field is $\bar{Z}^i$ and will be given to the direction coder along with N^i and M^i.

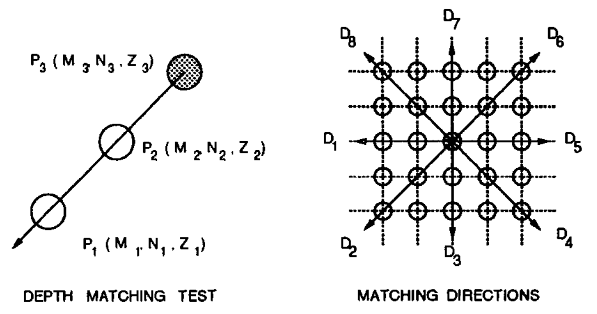


Figure 4: Determination of Direction Matching

4.3 Direction Encoder

Unmatched regions are usually from recently uncovered regions or highly curved regions that cannot be predicted. Since the pixels in those regions might come from some plane polygons or might be on the extension of a plane from a surrounding matched regions, a spatial prediction technique exploiting the plane relationship between neighboring pixels can be used to code these pixels. One such method is to find the matching direction in which two neighbor pixels lying on a straight line match the object number and depth of the current pixel and record the direction value by overwriting the $M^i_{m,n}$ which was originally zero. Since only the reconstructed data are available in the decoder, those two pixels should be pixels that have been already coded. Therefore it should be checked whether the matching status values of those two pixels is still zero. The matching conditions for a direction at P_3 in Fig. 4 are described as follows :

$$M_1 \neq 0 \text{ and } M_2 \neq 0 \tag{7}$$

$$N_1 = N_2 = N_3 \tag{8}$$

$$|Z_3 - \acute{Z}_3| < \epsilon \tag{9}$$

where $\acute{Z}_3$ is the spatially predicted depth of the pixel P_3 in the specified direction as in Eq. 10.

$$\acute{Z}_3 = 2Z_2 - Z_1 \tag{10}$$

These conditions are tested for eight directions from direction D_1 to D_8 as in Fig. 4. The first matched direction becomes the matching direction of the pixel and the corresponding direction value, which is defined as $D_i = i + 1$ in Fig. 4, is assigned to $M^i_{m,n}$ which was originally zero. The depth of the current pixel is replaced by the predicted value in the matched direction as in Eq. 10 for consistency of the data in the encoder and decoder. If there is no matching direction, the number 10 is assigned, corresponding to a totally unmatched pixel. After direction coding, the frame data will be:

$$\bar{M}^i_{m,n} = \begin{cases} 1 & \text{matched} \\ 2..9 & \text{direction matched} \\ 10 & \text{totally unmatched} \end{cases} \tag{11}$$

$$\acute{Z}^i_{m,n} = \begin{cases} \tilde{Z}^i_{m,n} & \text{matched} \\ \acute{Z}^i_{m,n} & \text{direction matched} \\ Z^i_{m,n} & \text{totally unmatched} \end{cases} \tag{12}$$

N^i remains unchanged because the object number is losslessly coded.

Fig. 5 shows an example of direction coding where the region inside the polygon is originally unmatched. The encoding is performed from left to right and from bottom to top

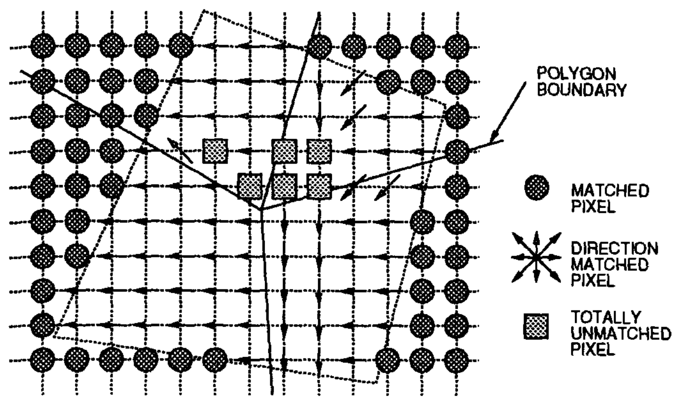


Figure 5: Example of Direction Encoding. The unmatched region inside the dashed polygon is coded by direction coding. Coding is performed from left to right and from bottom to top.

and the arrows represent the direction of matching. This illustrates that the number of totally unmatched pixels is very small and the matching directions are mostly 2 because direction 2 is the first test direction in Fig. 4. Since the geometrical data for the matched and direction matched pixels are predictable, the data to be transmitted are the matching status field $\bar{M}^i$ which has very low entropy and the $\{N^i_{m,n}, Z^i_{m,n}\}$'s for a few totally unmatched pixels.

4.4 Frame Synthesizer

Since the matched pixels can be identified from the matching status field $\bar{M}^i$ decoded in the receive decoder, the frame synthesizer can recover the geometrical data for all the matched pixels by copying the data from the predicted frame. Then the geometrical data $\hat{N}^i_{m,n}$, $\hat{Z}^i_{m,n}$ for the matched pixels and totally unmatched pixels are correctly recovered whereas those for direction matched pixels remain undetermined:

$$\hat{N}^i_{m,n} = \begin{cases} \tilde{N}^i_{m,n} = N^i_{m,n} & \text{if } \bar{M}^i_{m,n} = 10 \\ \tilde{N}^i_{m,n} = N^i_{m,n} & \text{if } \bar{M}^i_{m,n} = 1 \\ \text{undetermined} & \text{if } \bar{M}^i_{m,n} = 2..9 \end{cases} \quad (13)$$

$$\hat{Z}^i_{m,n} = \begin{cases} \tilde{Z}^i_{m,n} = Z^i_{m,n} & \text{if } \bar{M}^i_{m,n} = 10 \\ \tilde{Z}^i_{m,n} & \text{if } \bar{M}^i_{m,n} = 1 \\ \text{undetermined} & \text{if } \bar{M}^i_{m,n} = 2..9 \end{cases} \quad (14)$$

4.5 Direction Decoder

For the direction matched pixels, the object number $N^i_{m,n}$ is recovered by copying the object number of the pixel which is located in the matching direction and the depth $\hat{Z}^i_{m,n}$ becomes the predicted value $\acute{Z}^i_{m,n}$ from the two pixels located in the matching direction as in Eq. 10. $N^i_{m,n}$ are recovered correctly for all the pixels and the depth field $\hat{Z}^i$ is the same as $\acute{Z}^i$ in Eq. 12. The accuracies of $\tilde{Z}^i_{m,n}$ for the matched pixel and $\acute{Z}^i_{m,n}$ for the direction matched pixel are guaranteed by Eq. 6 and Eq. 9, respectively. These decoded data are fed into the predictor and are the same for both the encoder and decoder to make the same predictions as in Fig. 2.

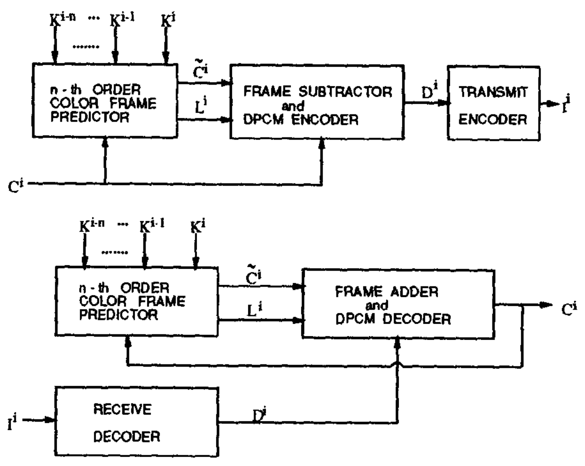

K^{i-n} ... K^{i-1} K^i
n-th ORDER COLOR FRAME PREDICTOR
$\tilde{C}^i$
L^i
FRAME SUBTRACTOR and DPCM ENCODER
D^i
TRANSMIT ENCODER
I^i
C^i
K^{i-n} ... K^{i-1} K^i
n-th ORDER COLOR FRAME PREDICTOR
$\tilde{C}^i$
L^i
FRAME ADDER and DPCM DECODER
C^i
I^i
RECEIVE DECODER
D^i

Figure 6: Color Data Encoder and Decoder, where $K^i = (T^i N^i \hat{Z}^i)$

5 Color Data Coding

Since the (T, N, Z)'s from the geometrical data coding block can be used to compute the locations of a current frame pixel on the previous frames and the previous color data frames are stored, the color of the pixel can be predicted by estimating the color at the transformed locations in the previous frames. The error image between the original and the predicted frame, also called the residual image R^i, has a relatively low entropy compared to the original. The pixels are classified into two classes, matched and unmatched pixels, based on whether the locations in the old frames are traceable or not. Since this residual image still has some spatial correlation, spatial linear prediction coding (DPCM) can be applied to reduce the entropy. This DPCM coded residual image is called a differential residual D^i and entropy coding using Huffman coding or arithmetic coding, can compress the differential residual losslessly. In decoder after the residual R^i is recovered by DPCM decoding, the original color data $C^i_{m,n}$ is obtained as the sum of the predicted value $\tilde{C}^i_{m,n}$ and the residual $R^i_{m,n}$ for the matched pixel. The color $C^i_{m,n}$ of the unmatched pixel is the same as residual $R^i_{m,n}$.

5.1 Color Frame Predictor

A pixel of one frame can be mapped to another frame by the transformation in Eq. 1 and if the transformed point satisfies the following condition, it is considered to be the same point as the current pixel and the pixel is said to be matched. If the pixel $P^i_{m,n}$ is transformed into the j-th frame and the transformed point is inside the pixel square $S^j_{p,q}$, the matching condition is as follows:

$$Z^j_{min} \le Z^j_{trans} \le Z^j_{max} \quad (15)$$

where the Z^j_{trans} is the depth of the transformed point and Z^j_{min}, Z^j_{max} are the minimum and maximum depth of the four corner pixels of $S^j_{p,q}$, respectively. Since several previous frames of geometrical and color data are available to deal with the occlusion problem, the transformed point in the nearest frame which satisfies the above condition is used to predict the color data of the current pixel.

L^i records the matching status values for color data which are zeros for the unmatched and ones for the matched pixels. Since generally the transformed point is not the pixel point, an interpolation is necessary for the computation of the color.

5.2 Subtractor and DPCM Encoder

Colors of matched regions in the residual R^i are residual values generated by sutracting the predicticed colors from the original colors whereas the colors in the unmatched regions remain unchanged as follows:

$$R^i_{m,n} = \begin{cases} C^i_{m,n} & \text{if } L^i_{m,n} = 0 \\ C^i_{m,n} - \tilde{C}^i_{m,n} & \text{if } L^i_{m,n} = 1 \end{cases} \tag{16}$$

Unmatched regions are mostly from recently uncovered regions and the entropy of such unmatched regions can be reduced by spatial linear predictive coding (DPCM). The residual image in the matched region has low entropy caused by changes of illumination, by the movement of objects, viewpoints or light sources between frames. Since this kind of error has relatively slow spatial variation, DPCM can be effectively applied also to the matched regions of the residual image.

These two steps, frame subtraction and DPCM, can be implemented with a combined operation as in Fig. 6. Since the matched and unmatched regions have different kind of data as explained above, each region should be coded independently. Usually 2-D DPCM uses three left and lower neighboring pixels $\acute{C}_{p,q}$ to predict the current pixel $C^i_{m,n}$. If the current pixel is an unmatched pixel, $\acute{C}^i_{p,q}$ should be the original color $C^i_{p,q}$. If the current pixel is a matched pixel, $\acute{C}^i_{p,q}$ should be the residual value. Since the residual value is not available for an unmatched pixel, the $\acute{C}^i_{p,q}$ is set to zero when $L^i_{p,q} = 0$ as follows:

$$\acute{C}^i_{p,q} = \begin{cases} C^i_{p,q} & \text{if } L^i_{m,n} = 0 \\ L^i_{p,q}(C^i_{p,q} - \tilde{C}^i_{p,q}) & \text{if } L^i_{m,n} = 1 \end{cases} \tag{17}$$

The spatially predicted value $\dot{C}^i_{m,n}$ is the integer part of a linear combination of those as follows:

$$\dot{C}^i_{m,n} = \text{int}(\alpha\acute{C}^i_{m-1,n} + \beta\acute{C}^i_{m-1,n-1} + \gamma\acute{C}^i_{m,n-1}) \tag{18}$$

In this paper, the prediction coefficients α, β, γ are selected to be 0.75,-0.5,0.75 respectively.

Then the differential residual image D^i is obtained by subtracting the spatially predicted value $\dot{C}^i_{m,n}$ from the residual value as follows:

$$D^i_{m,n} = \begin{cases} C^i_{m,n} - \dot{C}^i_{m,n} & \text{if } L^i_{m,n} = 0 \\ C^i_{m,n} - \tilde{C}^i_{m,n} - \dot{C}^i_{m,n} & \text{if } L^i_{m,n} = 1 \end{cases} \tag{19}$$

The differential residual D^i, which has very small entropy, can be compressed losslessly by an entropy coding technique.

5.3 Adder and DPCM decoder

The $L^i_{m,n}$ and $\tilde{C}^i_{m,n}$ in the predictor of the decoder are the same as those in the encoder. The original color field C^i is recovered by the combined step of DPCM decoding and frame addition as follows:

$$C^i_{m,n} = \begin{cases} D^i_{m,n} + \dot{C}^i_{m,n} & \text{if } L^i_{m,n} = 0 \\ D^i_{m,n} + \tilde{C}^i_{m,n} + \dot{C}^i_{m,n} & \text{if } L^i_{m,n} = 1 \end{cases} \tag{20}$$

where the spatial prediction $\dot{C}^i$ is the same as in the encoder. The recovered color C^i is fed back to the predictor for the prediction of the next frames.

6 Test Results

31 animation frames were generated to test the proposed compression algorithm. Each frame is composed of 7 objects and various kinds of textures were mapped by solid texture mapping. Through the whole 31 frames the ball bounces back and forth between the two wood blocks. In the first 11 frames the viewpoint does not change and in the next 10 frames the view point moves approximately 5 degrees/frame. During the last 10 frames zooming is performed. Plate 2,3, and 4 show the frames for above three cases.

In the case of Plate 2, since most of the objects are not moving and the view point is fixed, all regions except the rolling ball are matched regions. Pixel points in the current frame are transformed exactly to the same pixel points of other frames, then there are no errors due to bilinear color interpolation and since even the colors on the edges of stationary objects are predictable, the residual will have extremely small entropy which results in very high compression gain.

Plate 3 is a more general case in which both an object and the viewpoint are moving.There are mainly three kinds of residual errors. First, the recently uncovered regions are the major error regions which can be compressed only by DPCM. Second, in matched regions the changes in illuminations on the object surfaces causes residual errors. Usually illumination changes are due to the changes of specular reflection which varies with the movement of the view point or the object itself. Generally these errors change slowly and can be lowered by DPCM. Third, since the object boundaries are often unmatched regions which do not satisfy Eq. 15, the residual errors are large there. And even in the case where the pixels on the object boundary are matched, relatively large errors due to the color interpolation occur because the color data of these pixels were generated by antialiasing. In Plate 4, since with zooming there are no recently uncovered regions and the spatial frequency is decreasing, the entropy of the residual signal will be smaller than that in Plate 3.

The second order compression algorithm was implemented and tested on this test sequence. Fig. 7 shows the entropies of the original pictures and the differential residual images by 2-D DPCM and motion prediction algorithm. Table 1 illustrates the entropies of several frames. Plates 5 and 6 show the differential residual images of the RED component by linear predictive coding (DPCM) and the new motion prediction, respectively. As explained above, the major errors in Plate 6 are on recently uncovered regions and along the object boundaries which are often unmatched regions. The biggest differences between DPCM and motion prediction occurred on the wood texture which has relatively higher spatial frequency than any other regions. In these high spatial frequency regions, motion prediction shows much higher performance than spatial linear predictive coding. Through the whole 31 frames, the entropy of the original frame is around 20.7 bits/pixel. In the first 11 frames where the viewpoint is fixed, only about 1 bits/pixel is required for the motion prediction technique, except for the first frame which cannot be motion predicted and is coded only by DPCM. This contrasts with DPCM which needs around 11.4 bits/pixel. The necessary bits/pixel for the next 10 frames in which both the viewpoint and object are moving is around 8.3 bits/pixel which is about 3 bits/pixel gain over DPCM.

	frame 5	frame 15	frame 25
$N_{ORIGINAL}$	20.88	20.78	20.58
N_{DPCM}	11.35	11.23	10.59
N_{GEOM}	0.32	1.06	0.93
N_{RGB}	0.67	7.20	6.19
N_{MOTION}	0.99	8.26	7.12

Table 1: Entropies of original image and residuals by DPCM and Motion Prediction (unit : bits/pixel)

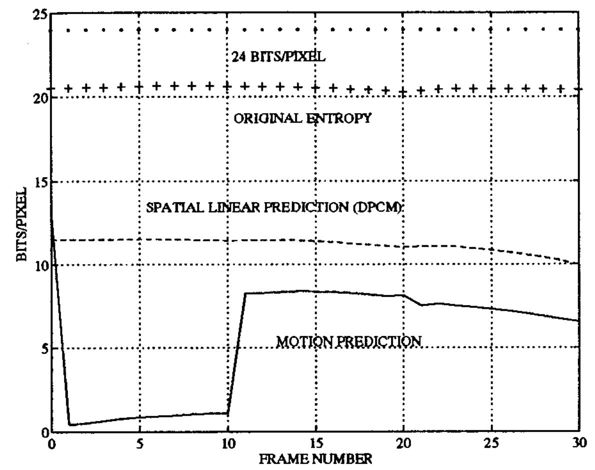


Figure 7: Entropies of the Bouncing Ball Sequence. The data for motion prediction are the sum of compressed geometrical data and entropy of residual.

In the next 10 frames, all entropies are decreasing with zooming as expected above, but the motion prediction algorithm still outperforms DPCM by about 3.5 bits/pixel.

7 Conclusion

The motion prediction compression algorithm for computer animation image sequences presented here consistently outperforms spatial linear prediction. This is true even though the new algorithm encodes double precision depth and integer object number information for a total of 96 bits/pixel including RGB data while the spatial compression algorithm does not. The depth and object number information are useful in their own right for performing z-buffer image compositing. The compression ratio achieved by the new algorithm was 1.4 times or more greater than that achieved with spatial linear prediction even for scenes with rapid changes in camera view point and substantial changes in object occlusion relationships. The overall compression ratio achieved with the new algorithm for image sequences with significant object and viewpoint motion was approximately 3 to 1.

There is still scope for improvement in the algorithm. We have noticed significant remaining correlation in the residual images. Long strings of zero residual values are punctuated by short bursts of plus and minus one residual values. Additionally the geometric overhead can be significantly reduced by storing the polygonal surface data and re-rendering it with the z- buffer algorithm, a process which should be no more time consuming than re-rendering pixel squares as in the current implementation. This makes the auxiliary data file more complex but for animation sequences more than a few seconds long the geometric data overhead should drop to .1 bit/pixel or less. We are currently implementing this extension.

The current implementation of the algorithm is limited to objects represented as polygonal surfaces. As discussed in the introduction this is not an undue restriction since efficient algorithms exist for approximating many different surface types with polygonal surfaces. However it would be an interesting research project to extend the geometric coder to other surface types so that an exact, rather than an approximating polygonal, surface representation could be used. This extension would require modifying both the motion prediction stage and the direction coding stage to compute surface equations directly from the depth information stored in the image.

8 Acknowledgments

The authors acknowledge help from Ragnar Jonsson, Stephen A. Martucci, Wayne Wooten and Lonnie D. Harvel.

References

[1] Denber, Michael J. and Turner, Paul M. A Differential Compiler for Computer Animation. Proceedings of SIGGRAPH '86 (Dallas, Taxas, August 18–22,1986). In *Computer Graphics* 20,4 (August 1986), 21–27.

[2] Duff, Tom. Compositing 3-D Rendered Images. Proceedings of SIGGRAPH '85 (San Francisco, California, July 22–26,1985). In *Computer Graphics* 19, 3(July 1985), 41–44.

[3] Jain, Anil K. *Fundamentals of Digital Image Processing.* Prentice-Hall, Englewood Cliffs, New Jersey, 1989.

[4] Jayant, Nuggehally S. and Noll, Peter. *Digital Coding of Waveforms.* Prentice-Hall, Englewood Cliffs, New Jersey, 1984.

[5] Jones, Stephen C. and Moorhead II, Robert J. Hardware-specific Image Compression Techniques for the Animation of CFD data. Proceedings of SPIE - International Society for Optical Engineering, 1668 (San Jose, California, February 10–11,1992), 141–146.

[6] Lim, Jae S. *Two-Dimensional Signal and Image Processing.*, Prentice-Hall, Englewood Cliffs, New Jersey, 1990.

[7] Martucci, Stephen A. Reversible Compression of HDTV Images Using Median Adaptive Prediction and Arithmetic Coding. Proceedings of IEEE ISCAS, 2(New Orleans, Louisiana, May 1–3,1990),1310–1313.

[8] Melnychuck, Paul W. and Rabbani, Majid. Survey of Lossless Image Coding Techniques. Proceedings of SPIE - International Society for Optical Engineering, 1075 (Los Angeles, California, January 17–20, 1989), 92–100.

[9] Sederberg, Thomas W. and Parry, Scott R. Free-Form Deformation of Solid Geometric Models. Proceedings of SIGGRAPH '86 (Dallas, Taxas,August 18–22,1986). In *Computer Graphics* 20,4 (August 1986), 151–160.

[10] Witten, Ian H., Neal, Radford M., and Cleary, John G. Arithmetic Coding for Data Compression. In *Communications of the ACM* 30,6 (June 1987), 520–540.

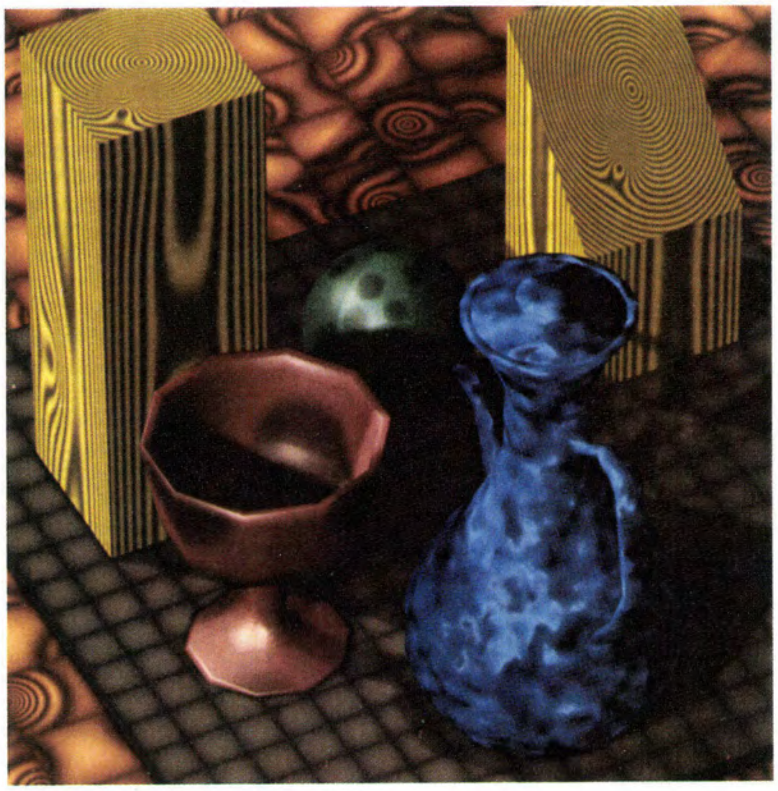


Plate 1: Frame 5 of the test sequence

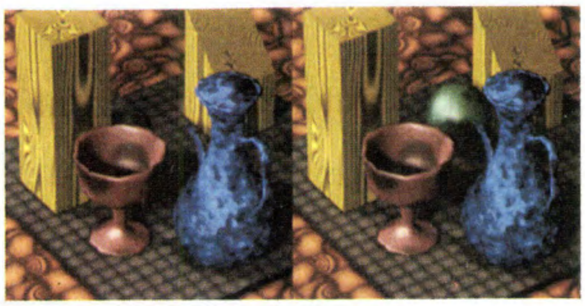


Plate 2: Frame 0,9

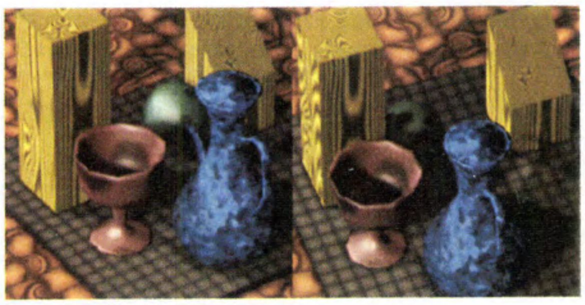


Plate 3: Frame 10,19

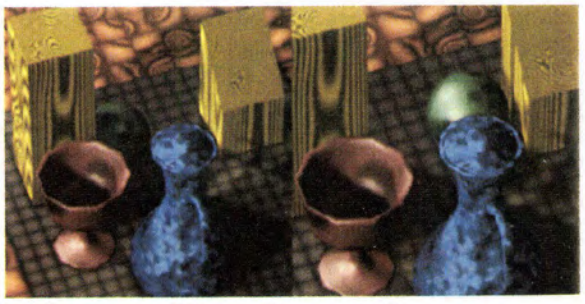


Plate 4: Frame 20,29

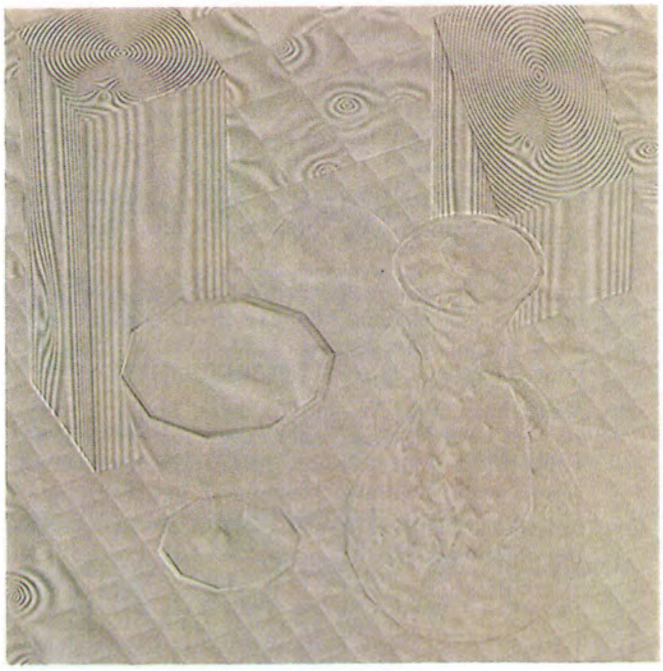
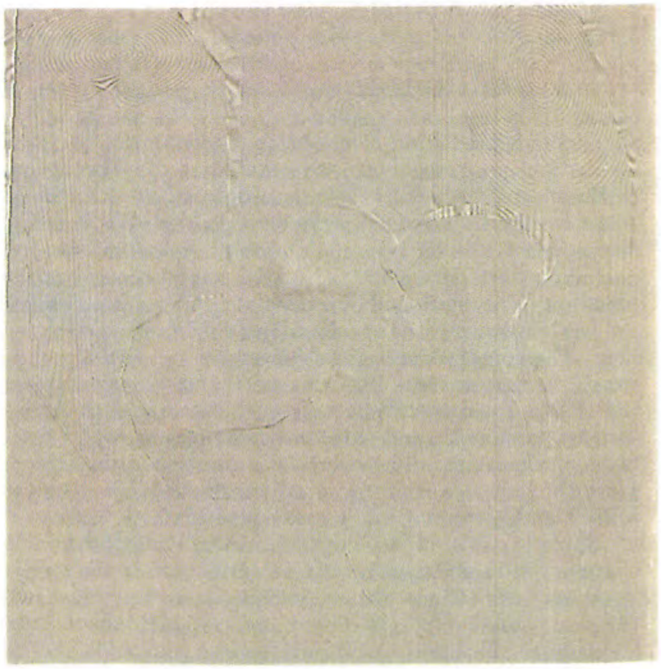



Plate 5: The differential image of frame 5 by DPCM
(512×512).

Space Diffusion: An Improved Parallel Halftoning Technique Using Space-Filling Curves

Yuefeng Zhang and Robert. E. Webber
Computer Science Department
The University of Western Ontario
London, Ontario, Canada N6A 5B7

ABSTRACT

Dot diffusion has been proposed as a way of combining the strengths of ordered dithering and error diffusion to create a parallelizable halftoning technique. However, dot diffusion pays a price in image quality in order to achieve parallelizability. Space-filling curves have been used to improve error diffusion. We show that by combining dot diffusion with a space-filling curve traversal technique, a parallelizable halftoning technique results that does not pay a cost in image quality. This new technique we call *space diffusion*.

KEYWORDS: Space-filling curves, dot diffusion, error diffusion, ordered dithering, digital halftoning, parallel algorithms.

1 INTRODUCTION

1.1 OVERVIEW

To print a gray scale image on a binary (e.g., black and white) device such as a laser printer, the gray image has to be transformed into a binary image. The process of doing this while not significantly altering the appearance of the image is called digital halftoning. Although grey-scale display devices are becoming cheaper and cheaper, there seems to consistently be a trade-off between the spatial resolution of display devices and the range of usable intensities at each spatial location. Thus it is reasonable to expect that for quite some time into the future, there will continue to be situations where the digital halftoning process will be necessary.

As the spatial resolution of display devices increases, the parallelizability of digital halftoning algorithms will play a greater role in the evaluation of the usefulness of a particular method in the design of an effective display system. Many digital halftoning methods have been proposed in the literature. The ones we will consider in this paper are: ordered dithering, error diffusion, dot diffusion, error diffusion along space-filling curves, and error diffusion along space-filling curves combined with patterning. Other methods may be found elsewhere [3, 5, 7]. We will propose a new method, called space diffusion, which combines the best features of these methods while remaining parallelizable.

In the following subsection, the above mentioned previously known halftoning methods are presented in more detail. After that, the space diffusion method is presented in Section 2. Then, Section 3 contains a discussion of experimental results aimed at evaluating the performance of this method. Final remarks are contained in Section 4.

For convenience, in the following discussion, we assume that the gray image and its corresponding binary image have the same size.

1.2 BACKGROUND

Of the previously known halftoning methods, ordered dithering [3] is one of the more widely used digital halftoning techniques. It is based on the idea of mapping grey values to black and white by comparing them to a threshold value that varies across the image. Suitable patterns of thresholds map regions of similar grey values into corresponding patterns of black and white values. Because the thresholds are independent of the image values and the mapping for each image value is independent of all the other image values, the method is very easily parallelized. The major drawback of this method is that it tends to produce regular patterning in large uniform regions as shown later in Section 3 where experimental results are presented.

This drawback is avoided in Floyd and Steinberg's error diffusion technique [2]. The error diffusion method is based on using a single threshold that is the same across the whole image. Instead of ignoring the difference between the actual value and the result of the thresholding (as ordered dithering does), error diffusion takes this difference and passes it on to neighboring nodes. This process is performed left-to-right row-by-row, so the error is spread (diffused) over the values in front and below the value currently being processed. This method is inherently non-parallel as it is based on a very specific order of processing of the individual grey values.

The error diffusion technique usually produces images that are much sharper than the images produced by the ordered dither methods. One difficulty with this method is that it sometimes generates features that do not appear in the original image. These features are the result of the distribution of error values across the image and hence do not occur in ordered dithering where the error values are not distributed at all.

Knuth [6] presents a new technique, called dot diffusion, that inherits the main advantages of ordered dithering and error diffusion techniques. At the same time, it avoids some of the difficulties with these two methods. The strategy of dot diffusion is to replicate a fixed matrix over the image. This assignment of matrix values to image locations divides the pixels of the image into classes that map to the same matrix value. Then, a technique similar to error diffusion is used to halftone these classes simultaneously. Specifically, the following matrix

$$\begin{pmatrix} 34 & 48 & 40 & 32 & 29 & 15 & 23 & 31 \\ 42 & 58 & 56 & 53 & 21 & 5 & 7 & 10 \\ 50 & 62 & 61 & 45 & 13 & 1 & 2 & 18 \\ 38 & 46 & 54 & 37 & 25 & 17 & 9 & 26 \\ 28 & 14 & 22 & 30 & 35 & 49 & 41 & 33 \\ 20 & 4 & 6 & 11 & 43 & 59 & 57 & 52 \\ 12 & 0 & 3 & 19 & 51 & 63 & 60 & 44 \\ 24 & 16 & 8 & 27 & 39 & 47 & 55 & 36 \end{pmatrix}$$

is usually used to classify the pixels of the image into 64 classes according to $(x \; mod \; 8, \; y \; mod \; 8)$, where (x, y) is the location of a pixel.

Starting with the pixels that map to the index value 0, at each stage, the thresholded value of pixels in the lowest numbered unprocessed class is computed and the resulting error difference is distributed to the remaining neighboring pixels that are in unprocessed classes with higher matrix numbers. As with error diffusion, the threshold used is the same at each location.

This method is related to error diffusion in that it also distributes the error term that correspond to the difference between the original grey value and the resulting thresholded binary value. It is related to ordered dithering in that ordered dithering is also generally implemented by way of a replicated matrix that establishes a positional bias in location of black and white values in the image. This matrix replication structure causes bother ordered dithering and dot diffusion to suffer from the problem of having regular patterning occur in large regions of uniform intensity as illustrated in Section 3. Fixing this problem with dot diffusion is one of the main aspects of this paper.

Recently [1, 8, 9, 10, 11], it has been observed that error diffusion itself can be improved by replacing the left-to-right row-by-row order of processing of pixels by an order patterned after the discrete version of space filling curves. Two commonly used curves are Peano curves [1] and Hilbert curves [8]. Our method is based on Hilbert curves and they are discussed in more detail in Section 2.

The main idea of the halftoning techniques that are based on space-filling curves is to use the space-filling curves to establish the order in which the pixels of the image are visited. Error terms are then distributed along this path in a manner similar to error diffusion.

One problem with using traditional space-filling curves for traversal patterns is that they fit best on square images that have widths that are powers of two. However, two approaches exist in the literature to addressing this problem for halftoning algorithms. One approach is based on a generalization of Peano curves called Murray polygons [1]. Another approach exists based on decomposing rectangles into smaller rectangles from one of a few standard sizes [11].

However, these methods have two other drawbacks. One is that, like traditional error diffusion, they are inherently serial (as the algorithm is driven by a particular serial traversal of the pixels). The other is that they do not handle long narrow features in images as well as either dot diffusion or error diffusion do. On the other hand, this method does handle large uniform regions in images better than dot diffusion or ordered dithering. These aspects of the method are illustrated in Section 3.

A further refinement of the space-filling traversal method is to use small standard patterns to replace segments of the halftoned image [8]. This can be used to compensate for some problems common to laser printing involving interaction between neighboring 'dots' on a page of paper. This method was also investigated in Section 3. Its main drawback is that it blurs the image.

In this paper, we propose a parallel mechanism for the halftoning techniques using space-filling curves. Specifically, discrete Hilbert curves are used to subdivide the image into regions. Other space-filling curves could be used as well. A strategy similar to dot diffusion is used to halftone the resulting regions simultaneously. This method avoids the difficulties with both the dot diffusion and the above space-filling curve halftoning techniques.

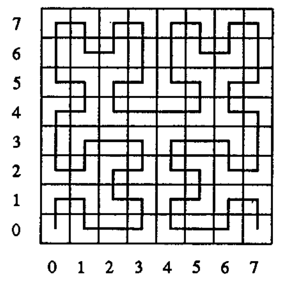
2 SPACE DIFFUSION

In our method, the gray image is subdivided into small regions along a space-filling curve. Then, the order of pixels along the segment that traverses a given region is used to assign the pixels of that region to their appropriate classes. Finally, a strategy similar to the dot diffusion technique is used to halftone these classes of pixels simultaneously.

Figure 1a shows a discrete Hilbert curve covering an 8×8 square region. Such curves can be easily generated to cover any $2^n \times 2^n$ region. The particular curve in Figure 1a defines a traversal ordering of the 8×8 image as shown in Figure 1b. This ordering is derived by labeling the pixel at one end of the curve by 0 and then incrementally labelling each of the remaining pixels as one moves along the curve until the last pixel is reached which is labeled 63.

This image can be subdivided into 4×4 regions by replacing each label by its value mod16 (4×4) as shown in Figure 1c. In general, we could break any $2^n \times 2^n$ image up into $2^m \times 2^m$ regions by first labeling each pixel of the region by its Hilbert curve ordering and then reducing these labels (mod 2^{2m}). Within each of these regions, the pixels are assigned classes whose numbers are integer values from 0 to 2^{2m}. These class assignments can then be used as the basis of a dot diffusion process. However, it differs from the traditional dot diffusion setup in that the matrices are not all oriented in the same manner. Figure 1d shows what the class assignments might be if we had attempted to replicate the initial 4×4 matrix of Figure 1c, rather than allowing the Hilbert curve to re-orient each of the matrices.

The class assignments shown in Figure 1d can be viewed as the basis of a standard dot diffusion process using a slightly different base matrix. Indeed, if we were breaking an image into 8×8 regions, then we would end up with matrices similar to that of Figure 1b. If we compare that matrix of class assignments to those in the matrix given for dot diffusion in Section 1, we see that matrix defined by the space filling curve does not.distribute the classes as well as the matrix given in the original presentation of the dot diffusion method [6]. However, the defects of this matrix are



(a)

7	21	22	25	26	37	38	41	42
6	20	23	24	27	36	39	40	43
5	19	18	29	28	35	34	45	44
4	16	17	30	31	32	33	46	47
3	15	12	11	10	53	52	51	48
2	14	13	8	9	54	55	50	49
1	1	2	7	6	57	56	61	62
0	0	3	4	5	58	59	60	63
	0	1	2	3	4	5	6	7

(b)

7	5	6	9	10	5	6	9	10
6	4	7	8	11	4	7	8	11
5	3	2	13	12	3	2	13	12
4	0	1	14	15	0	1	14	15
3	15	12	11	10	5	4	3	0
2	14	13	8	9	6	7	2	1
1	1	2	7	6	9	8	13	14
0	0	3	4	5	10	11	12	15
	0	1	2	3	4	5	6	7

(c)

7	15	12	11	10	15	12	11	10
6	14	13	8	9	14	13	8	9
5	1	2	7	6	1	2	7	6
4	0	3	4	5	0	3	4	5
3	15	12	11	10	15	12	11	10
2	14	13	8	9	14	13	8	9
1	1	2	7	6	1	2	7	6
0	0	3	4	5	0	3	4	5
	0	1	2	3	4	5	6	7

(d)

Figure 1: A 8x8 Hilbert polygon and its corresponding subdivisions.

more than compensated for by the result of constantly re-orienting it along a space-filling path as illustrated by the experimental results in Section 3. It is worthwhile noting that determining a good matrix when the orientation of the matrix is kept constantly the same was a particularly difficult aspect of the development of the original dot diffusion method.

The method based on this restructuring of the matrix orientations we call *space diffusion*. Although the example in Figure 1 presents the method in terms of square matrices being re-oriented, unlike dot diffusion, there is no need to break the image up into square regions. We can make the class assignments by reducing the original Hilbert labels by mod k for any value of k rather than restricting ourselves to values of k that are squares of powers of 2. However, in using these odd shaped decompositions, we haven't observed any significant differences in image quality from when we used squares of powers of two as the basis for class assignments.

The details of error distribution are exactly the same as those presented in Knuth's paper [6]. As with dot diffusion, the error distribution process is inherently parallel since information does not pass between pixels in different regions.

3 EXPERIMENTAL RESULTS

By combining space filling traversals with diffusion of the error term, we hope to gain the sharper detail characteristic of methods such as error diffusion and dot diffusion while taken advantage of other space filling methods' superior handling of large uniform regions (i.e., areas lacking sharp details).

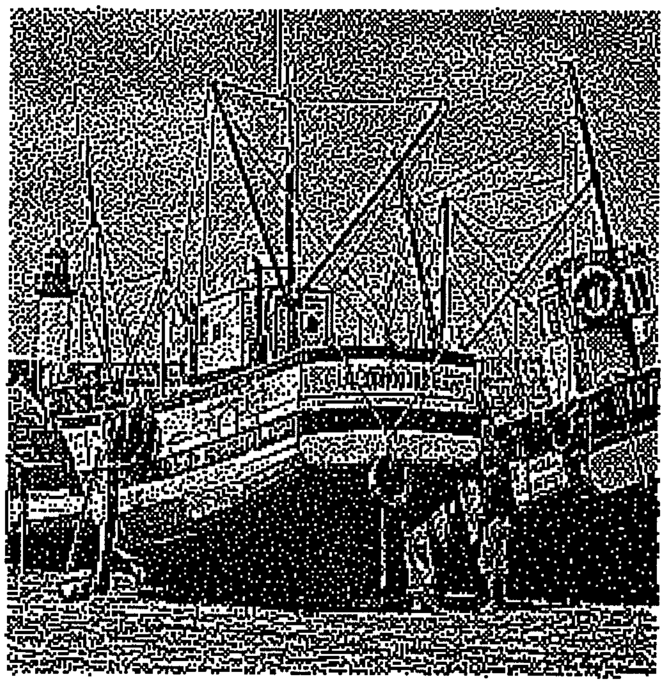


Figure 2: Space diffusion method on enhanced image of boat scene

However, to know if this goal is actually accomplished, it is necessary to evaluate the method on actual grey images.

Figures 2 through 13 show a good image for comparing the various methods in that these figures combine both finely detailed regions with large uniform regions. The scene of these figures is a couple of boats in front of a light house. The rigging of the boats provides the fine detail and the hull of the boat provides the main example of a large uniform region although the sky is also of interest in this regard. The image is 256 × 256. Traditional edge enhancement has been applied to the original image since that improves the results of all methods in Figures 2 through 7. Figures 8 through 13 illustrate the results on the un-enhanced version of the same images. In the un-enhanced images, most of the fine detail is lost by all methods.

Figures 2 through 7 illustrate each of the 6 digital halftoning techniques described in this paper. Figure 2 was done using the space diffusion method described in Section 2 where the number of distinct classes assigned was 36. The significant fine detail to observe occurs in two places: 1) the rigging to the immediate right of the lighthouse (which is itself midway down the left-hand side of the image) and 2) the curved rigging immediately above the cabin of the boat on the right-hand side of the image.

Figure 3 was done using dot diffusion. Figure 4 was done using error diffusion. Both of these images preserve this fine detail. However, Figure 5, which was done using the traditional space filling curve, can be seen to have lost the sharpness of the details in these areas. Figure 6 was done using the pattern based version of the space filling method and shows significant loss of fine detail. Figure 7 was done by ordered dithering and seems to have a similar loss of fine detail to the traditional space filling method.

Looking at the large uniform areas in each of these images, we observe that the ships hull in the dot diffusion im-

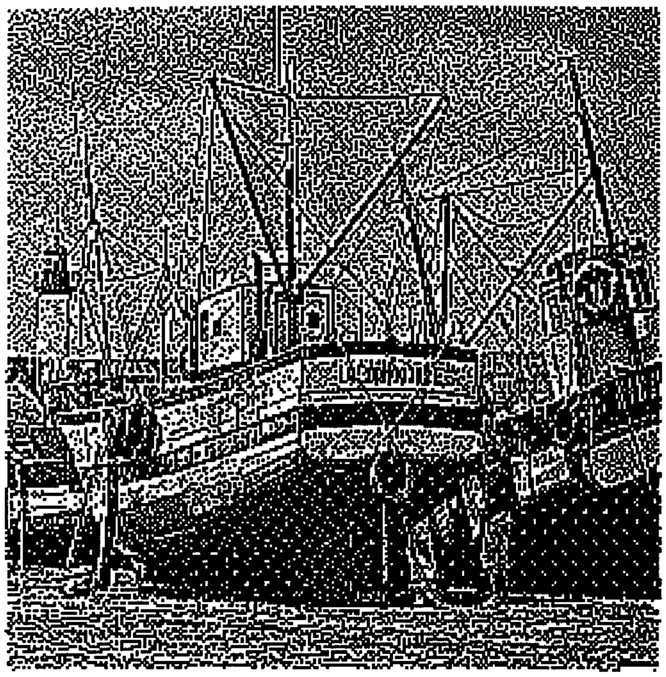


Figure 3: Dot diffusion method on enhanced image of boat scene

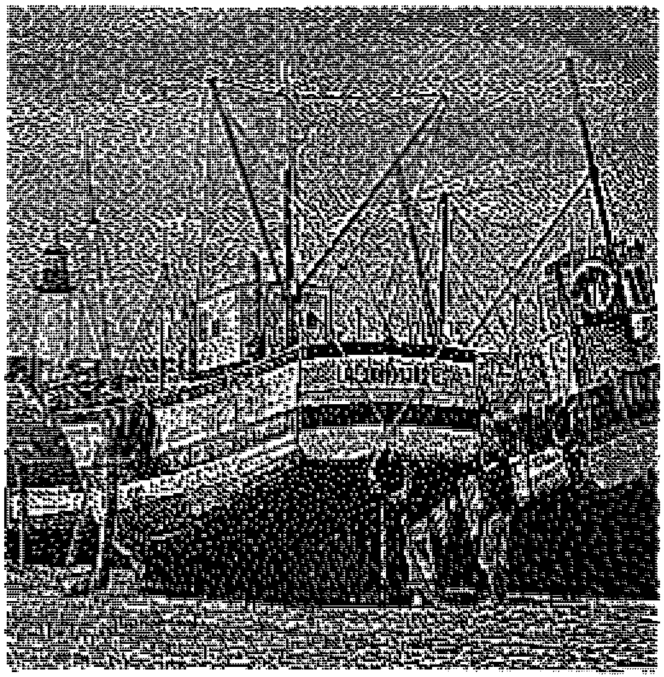


Figure 5: Space-filling curve method using run summation on enhanced image of boat scene

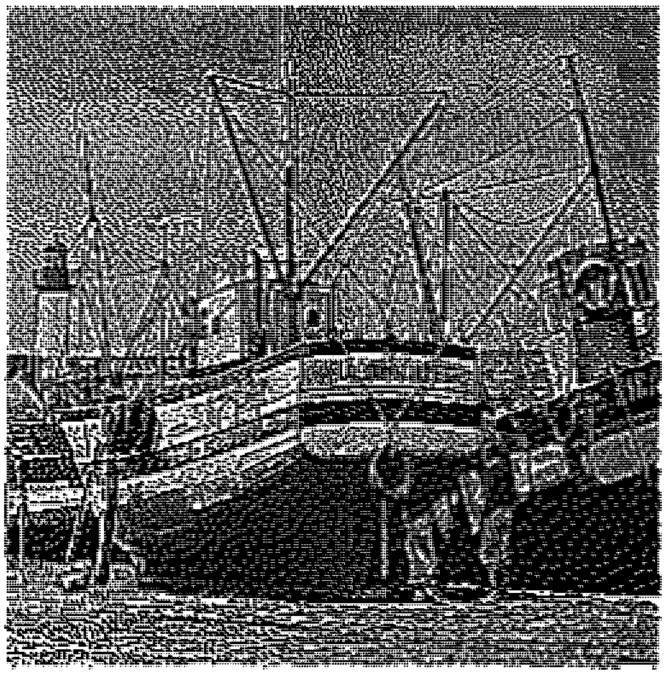


Figure 4: Error diffusion method on enhanced image of boat scene

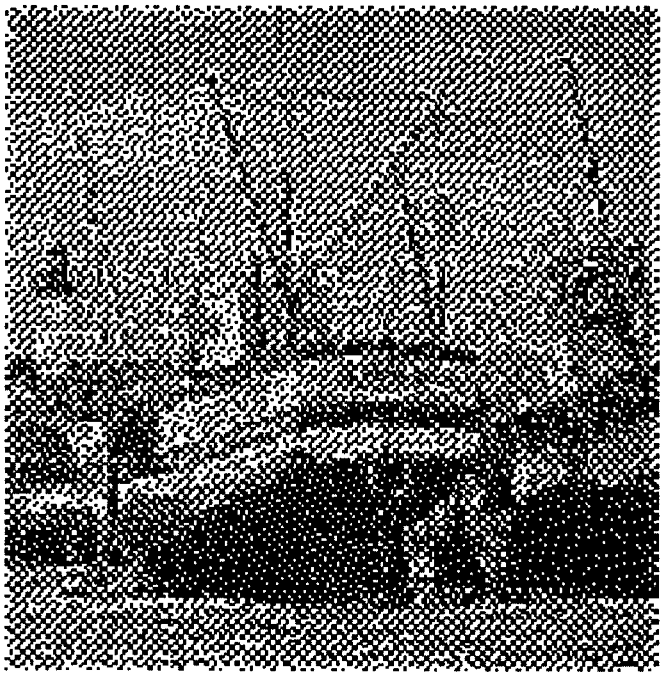


Figure 6: Space-filling curve method using patterns of up to 8 pixels on enhanced image of boat scene

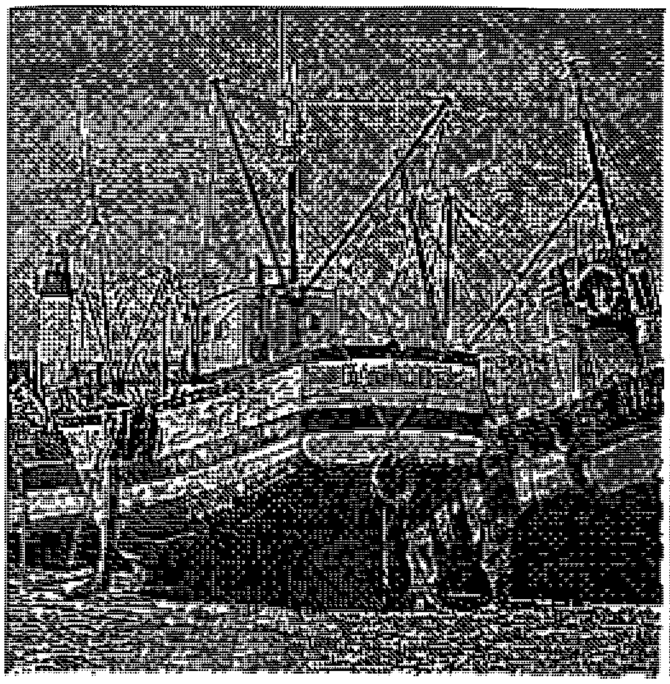


Figure 7: Ordered dithering method using 8x8 patterns on enhanced image of boat scene

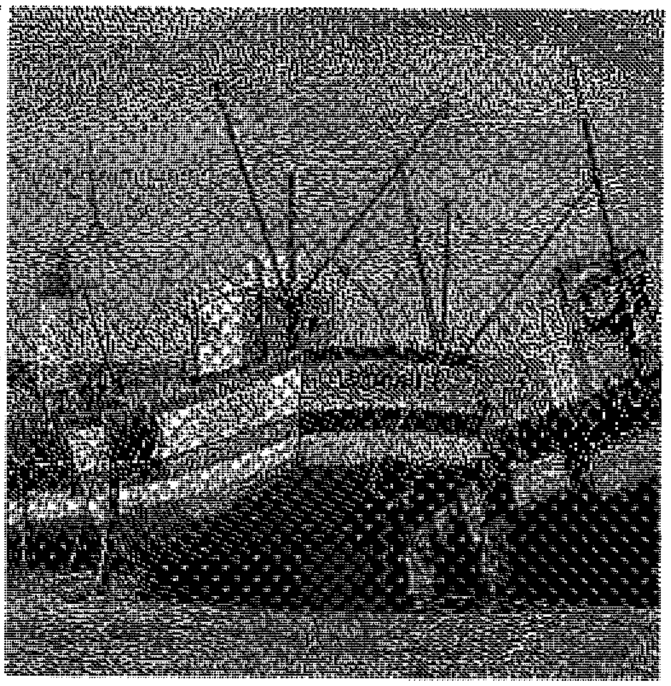


Figure 9: Dot diffusion method on un-enhanced image of boat scene

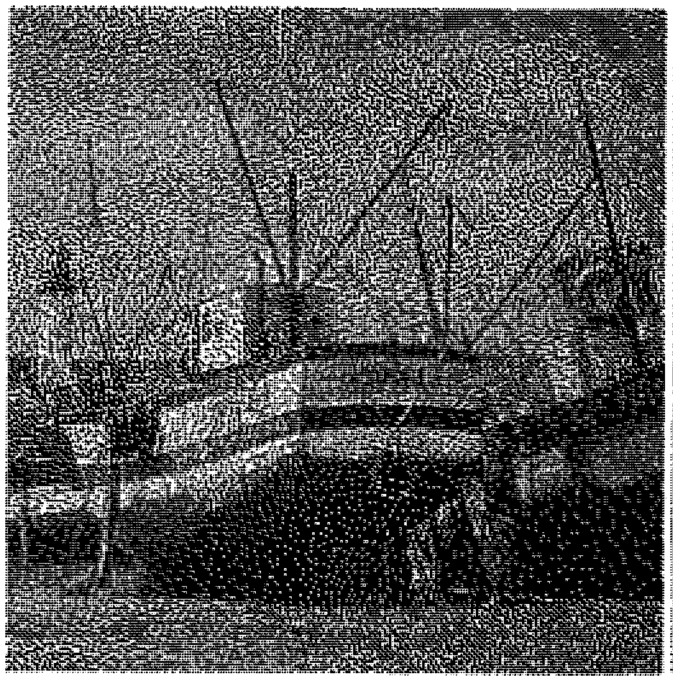


Figure 8: Space diffusion method on un-enhanced image of boat scene

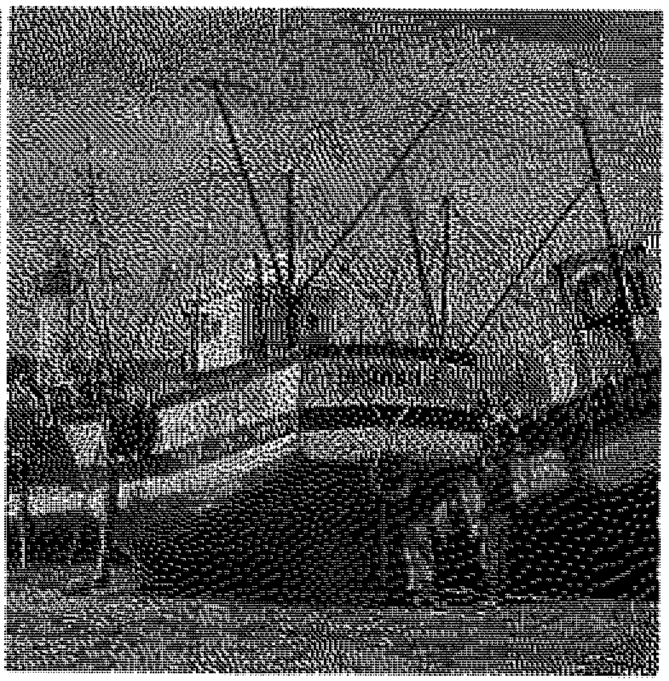


Figure 10: Error diffusion method on un-enhanced image of boat scene

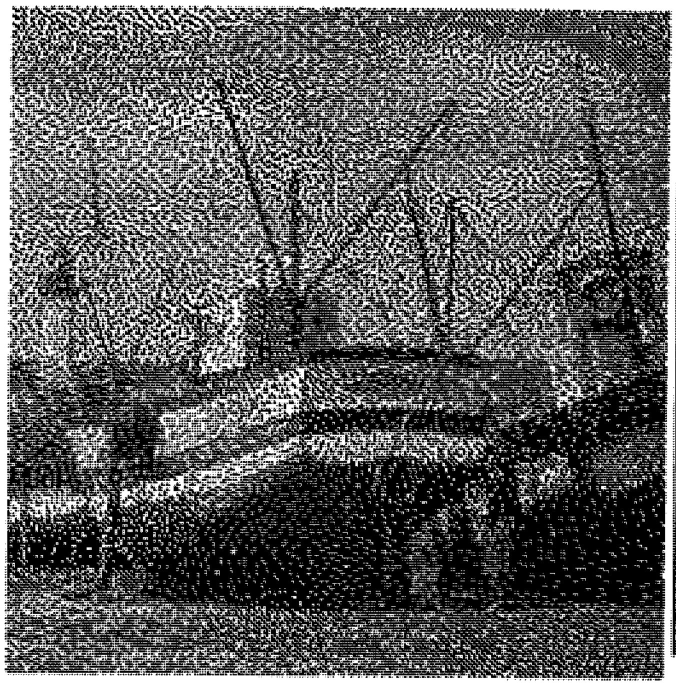


Figure 11: Space-filling curve method using run summation on un-enhanced image of boat scene

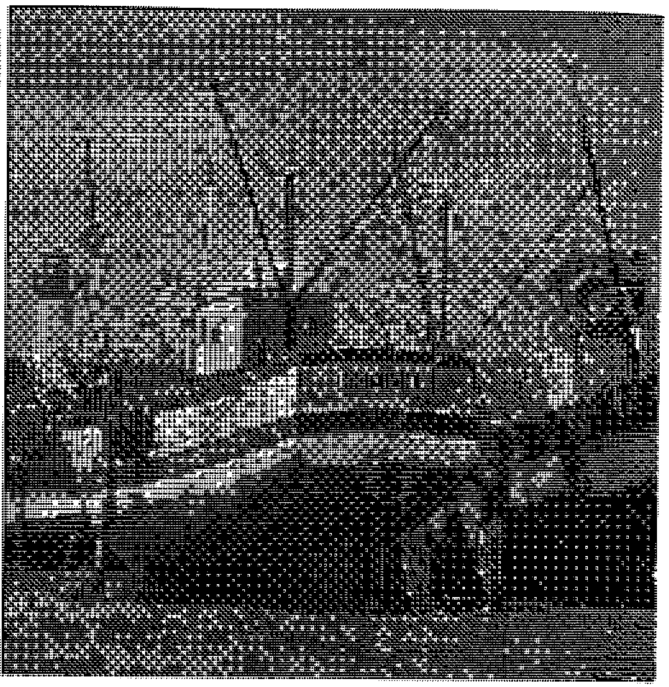


Figure 13: Ordered dithering method using 8x8 patterns on un-enhanced image of boat scene

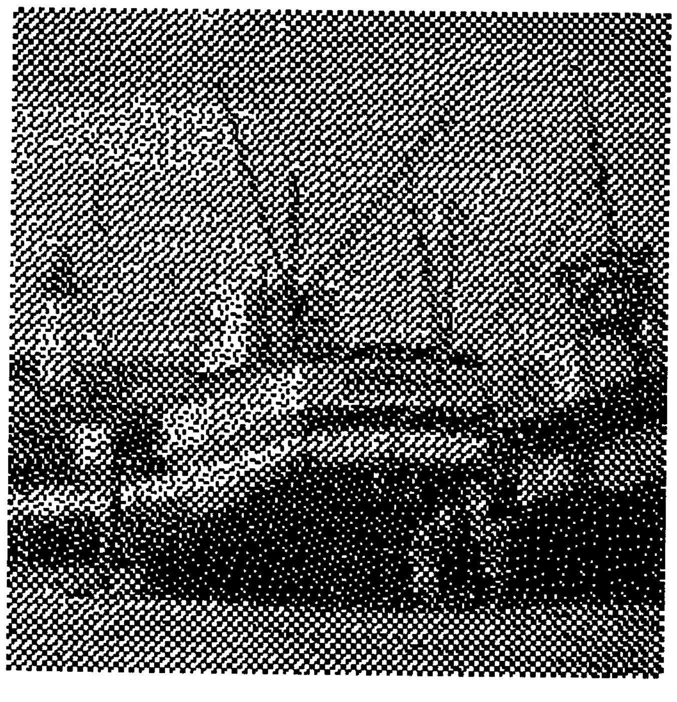


Figure 12: Space-filling curve method using patterns of up to 8 pixels on un-enhanced image of boat scene

age and in the ordered dither image show significant regular patterning. In the error diffusion method, there appears to be a fairly regular grain to the patterning underlying the sky portion of the image. This patterning is not evident in the space diffusion image. This kind of regular patterning is more evident in pictures with larger more uniform regions. Figure 14 shows such an image.

Figure 14 presents the same six methods as used to create the images of Figure 2 through 7, but this time the underlying grey image is a simple rectangle with a constantly increasing intensity moving from right to left. The top image was produced using space diffusion. The image immediately below it is the result of dot diffusion and the regular patterning is quite evident. Below that is an image produced by error diffusion. As one moves across this image, one encounters many grey regions where there is a regular bias in the patterning. Below the error diffusion image is the traditional space-filling curve method. On this image it produces very much the same result as the space diffusion method. Below the space-filling image is an image produced by ordered dithering that produces strikingly regular patterning artifacts. The bottom image was produced using the patterning version of the space-filling method. These small patterns produce very regular patterning at some of the grey intensity values (for example, near the middle of the rectangle).

In summary, experimental results indicate that the space diffusion method is similar in quality to both the error diffusion method and the traditional space-filling curve method. However, each of these methods as a slight weakness in comparison to the space diffusion method (the error diffusion method is worse on large uniform regions and the space-filling method is worse on fine detail). In addition to this, both of these methods are inherently serial whereas the space diffusion method is easily parallelizable.

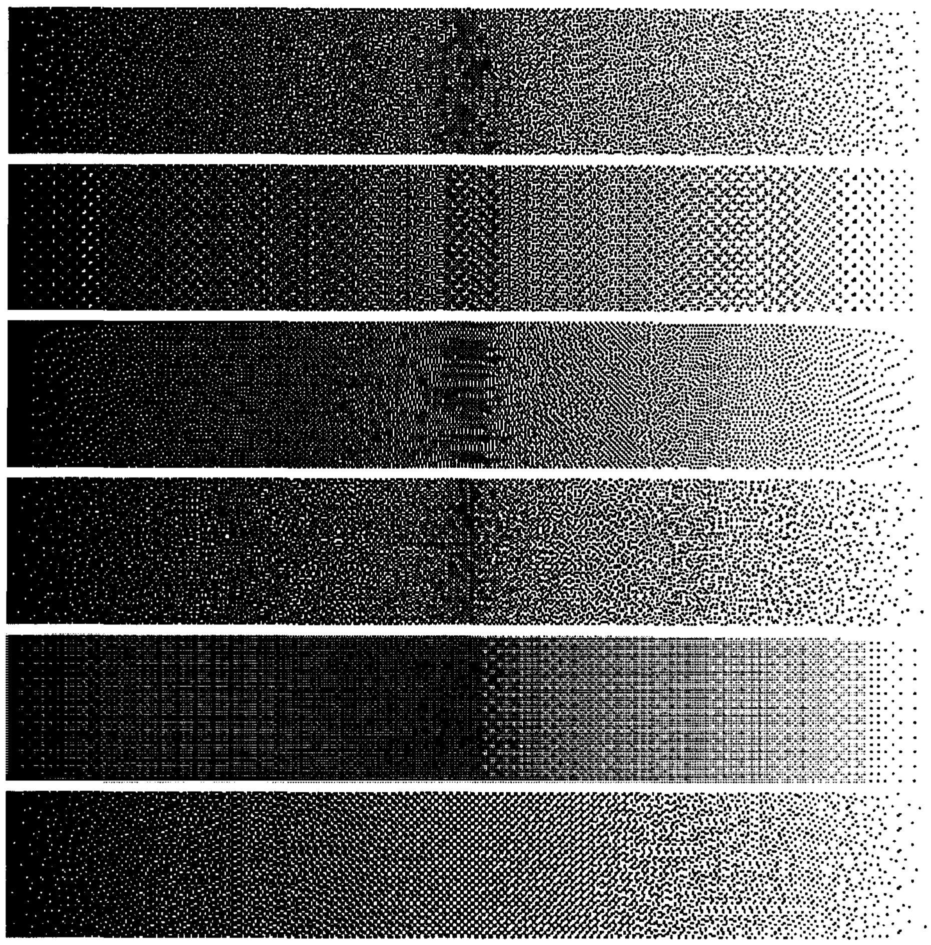


Figure 14: From top to bottom, the binary images are produced by: (1) space diffusion, (2) dot diffusion, (3) error diffusion, (4) space-filling curve method using run summation, (5) ordered dithering with 8 × 8 patterns, and (6) space-filling curve method using patterns of up to 8 pixels.

4 CONCLUSION

In this paper, we presented a new digital halftoning technique that subdivides the image into regions by subdividing a space-filling curve over the image into segments. The pixels of the image are then divided into classes by using coordinate translations on these segments. A strategy similar to dot diffusion is used to halftone these classes of pixels simultaneously.

So far, all the published halftoning techniques based on space-filling curves are sequential. This paper provides a parallel mechanism for these methods. The traditional dot diffusion method uses a fixed square matrix to divide the pixels of the image into classes. Our method, however, allows various subdivisions of the image by dividing the space-filling curve over the image into segments in different ways or using different space-filling curves. In this sense, our method may serve as a generalization of the traditional dot diffusion method.

In comparing various halftoning methods, the key problems appear to be loss of fine detail (or image sharpness) on the one hand and regular patterning in large uniform regions on the other hand. The space diffusion method appears to be better than other known methods in both these measures although the difference is not always large.

In general, we believe that space diffusion is a significantly better way of parallelizing error diffusion than is the dot diffusion method.

References

[1] Cole, A. J. Halftoning Without Dither or Edge Enhancement. *The Visual Computer* 7 (1991), 232 - 246.

[2] Floyd, R. W. and Steinberg, L. An Adaptive Algorithm for Spatial Greyscale. *Proceedings of the S. I. D.* 17, 2 (Second Quarter 1976), 75 - 77.

[3] Jarvis, J. F., Judice, C. N., and Ninke, W. H. A Survey of Techniques for the Display of Continuous Tone Pictures on Bilevel Displays. *Computer Graphics and Image Processing* 5 (1976), 13 - 17.

[4] Judice, C. N., Jarvis, J. F. and Ninke, W. H. Using Ordered Dither to Display Continuous Tone Pictures on an AC Plasma Panel. *Proceeding of the S.I.D.* 15, 4 (Fourth Quarter, 1974), 161 - 169.

[5] Knowlton, K. and Harmon, L. Computer-Produced Grey Scales. *Computer Graphics and Image Processing* 1 (1972), 1 - 20.

[6] Knuth, D. E. Digital Halftones by Dot Diffusion. *ACM Transactions on Graphics* 6, 4 (October 1987), 245 - 273.

[7] Ulichney, R. *Digital Halftoning.* The MIT Press, Cambridge, Massachusetts, 1988.

[8] Velho, L. and de M. Gomes, J. Digital Halftoning with Space Filling Curves. Proceedings of SIGGRAPH '91 (Las Vegas, 28 July-2 August 1991). In *Computer Graphics* 25, 4 (July 1991), 81 - 90.

[9] Witten, I. H. and Neal, M. Using Peano Curves for Bilevel Display of Continuous Tone Images. *IEEE Computer Graphics and Applications* (May 1982), 47 - 52.

[10] Witten, I. H. and Wyvill, B. On the Generation and Use of Space-filling Curves. *Software Practice and Experience* 13 (1983), 519 - 525.

[11] Wyvill, G. and McNaughton, C. Three Plus Five Makes Eight: A Simplified Approach to Halftoning. *Scientific Visualization of Physical Phenomena* (Boston, 1991), Springer-Verlag, New York, 379 - 392.

An Implicit Formulation for Precise Contact Modeling between Flexible Solids

Marie-Paule Gascuel*
iMAGIS, LIENS, CNRS URA 1327
Ecole Normale Supérieure, 45 rue d'Ulm
75005 Paris, France

Abstract

This paper presents an implicit deformable model, based on isosurfaces of potential fields generated by skeletons, that provides elegant and unified formulations for both geometric parameters such as shape or deformation and physical properties such as rigidity. The model is especially designed to improve collision and contact processing for non-rigid objects. In particular, it generates and maintains exact contact surfaces during interactions.

Keywords: animation, simulation, deformation, implicit surface, collision detection, collision response.

1 Introduction

Dynamic animation systems based on simplified physical laws have drawn a lot of attention during the past few years. One of the reasons why they seem so attractive is their ability to respond automatically to collisions. Nevertheless, contrary to rigid solid animation where complete analytical solutions have been found [1], modeling interactions between deformable objects still remains a challenge. In particular, none of the models proposed up to now generates an exact contact surface between interacting flexible solids.

This paper presents a new, continuous model for deformable material based on an implicit formulation which unifies the description of geometry and of physical properties of solids. Well adapted to the simulation of local deformations, the model is especially designed to improve collision and contact processing for non-rigid objects. In addition to an efficient collision detection mechanism, it generates and maintains exact contact surfaces during interactions. These surfaces are then used for the calculation of reaction forces.

Compact, efficient, easy to implement and to control, our implicit deformable model would be a particularly convenient tool in character animation where locally deformable flesh must be simulated.

*From september: iMAGIS, IMAG, BP53X 3841 Grenoble Cedex.

1.1 Previous approaches

Flexible models in Computer Graphics result from either nodal approaches (which include finite elements [6], finite differences [12, 13], and systems using elementary masses [8, 7]) or global approaches [10, 14]. The latter optimize the animation by approximating deformations by particular classes of global transformations. Well adapted to the animation of homogeneous blocks of elastic material, they would not, however, be convenient to use when simulating a material subject to local deformations (a sponge for instance), or when modeling non homogeneous complex objects like those used in character animation (typically, deformable coating over rigid skeletons).

Collisions between flexible objects are a complex phenomenon. In particular, they are not instantaneous and do not conserve energy. Among the solutions used to cope with this problem in Computer Graphics, penalty methods [9] are probably the most widely spread. They don't generate any contact surface between interacting flexible solids but use instead the amount of local interpenetration to find a force that pushes the objects apart. A different solution consists in using the relative stiffnesses of solids to find correct deformed shapes in contact situations. Here, response is computed by integrating deformation forces within the contact areas. But combined with a deformable model based on spline surfaces controlled by discrete spring systems [5], this method does not generate exact contact surfaces. A third approach [2] extends the analytical interaction processing used for rigid solids [1] to a global deformable model [14]. Contact surfaces are approximated by discrete sets of contact points which, as the authors emphasize, is somewhat unsatisfactory.

In all these methods, the lack of a contact surface between interacting flexible solids generates local interpenetration and imperfectly deformed shapes. The extent of these artifacts is exacerbated by the lasting quality of soft collisions, and forbids any correct evaluation of reaction forces.

1.2 Overview

This paper presents a new deformable model which improves interaction processing for flexible solids. Our main point is the use of isopotential implicit surfaces generated by "skeletons" to model the objects. Developed up to now as a tool for free form modeling [4, 3], this formalism has not been used as a way to model physical properties[1]. It leads to a concise formulation for both geometric parameters, such as shape and deformation, and physical properties, such as rigidity and elastic behavior. The deformable model is continuous and provides easy modeling of local deformations.

The associated method for collision detection and response is an improved version of [5]. The inside/outside functions associated with implicit solids greatly reduces the computational cost of collision detection. The model generates and maintains exact contact surfaces between interacting objects. Opposite compression forces are respectively applied to the solids along contact surfaces, so a correct integration of response forces can be calculated.

Section 2 describes the implicit deformable model, and explains how to design both homogeneous and non-homogeneous flexible solids. The processing of interactions is detailed in Section 3, including the particular cases of multiple collisions and of interactions with rigid solids. Section 4 discusses implementation. Section 5 focuses on the possibilities for future research opened by our method.

2 Implicit Deformable Solids

Our aim is to simulate damped material where deformations due to collisions remain local. Rather than considering the general Lagrange equations of motion for non-rigid objects (as in [12]), we use the same approximation as in [13, 5]: the mass distribution of solids is considered to be constant, so motion is calculated using rigid body equations which lead to a more efficiently computed animation. More precisely, deformable solids are split into two layers:

- A rigid component which obeys the rigid body equations of motion. Its mass distribution corresponds to the object's rest shape.
- A deformable layer at rest relative to the rigid layer.

This section presents a new model for the deformable layer based on implicitly defined isopotential surfaces generated by skeletons. We first review the definition of these surfaces.

2.1 Implicit surfaces

Implicit surfaces such as "distance surfaces" [4] and "convolution surfaces" [3] allow the free form design of shapes through the manipulation of "skeletons" that generate potential fields. Very simple to define and to control, they constitute a good alternative to traditional implicit surfaces defined by analytical equations. An implicit surface S generated by a set of skeletons $S_i (i = 1..n)$ with associated "field functions" f_i is defined by:

$$S = \{P \in \Re^3 \;/\; f(P) = 1\} \quad \text{where} \quad f(P) = \sum_{i=1}^{n} f_i(P)$$

[1]Flexible solids have been described by superquadrics [10, 11], another kind of implicit surface. Contrary to the approach developed here, the choice of an implicit geometric description of objects was not closely related to the way physical properties were modeled.

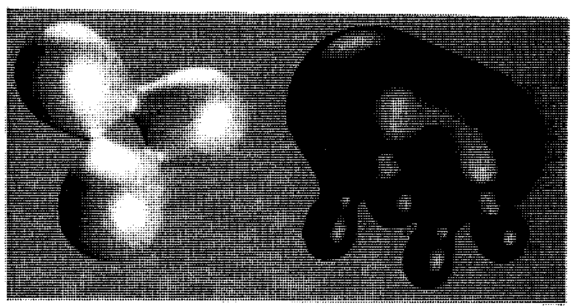


Figure 1: Isopotential objects generated by skeletons.

This surface surrounds the solid defined by $f(P) \geq 1$, which can have several disconnected components. Normal vectors are directed along the field's gradient.

The skeletons S_i can be any geometric primitive admitting a well defined distance function, such as: points, curves, parametric surfaces, or volumes. The field functions f_i are monotonically decreasing functions of the distance to the associated skeleton [4]. For convolution surfaces [3], they are given by integrals of exponential contributions from each point of the skeleton. In order to optimize the computations, these functions usually have a restricted scope of influence. Examples of isopotential surfaces are shown in Figure 1.

An implicit surface can easily be deformed by introducing a deformation term g in its implicit representation e.g. $f(P) + g(P) = 1$. In the remainder of this paper, only this type of deformations are considered.

2.2 Defining elastic material with potential fields

The method is based on the following observation: the set of points P satisfying $f(P) = 1$ (where f is the field function) is sufficient to define a surface. This set of points being fixed, the variation of f around the isosurface can be used to model physical properties. The next section explains how to express stiffness with field functions in a way which yields a very simple correspondence between applied forces and resulting deformations.

Correspondence between forces and deformations

A deformable model is defined by a correspondence between forces and deformations. In computer graphics, this correspondence has been given by both linear [13, 6, 14] and non-linear [12] elasticity. In non-linear models, the stiffness k is not only a function of the point P you consider, but may also depend on its current location inside the solid. The applied force during a displacement of P from $X_0 = (x_0, y_0, z_0)$ to $X(P) = (x(P), y(P), z(P))$ is:

$$R(P) = \int_{X_0}^{X(P)} k_P(Y)\, dY \qquad (1)$$

To improve generality, implicit deformable solids should be capable of exhibiting both linear and non-linear behaviors.

In practice, the correspondence between forces and deformations will be used during the collision process, to integrate the reaction forces colinear to normal vectors along contact surfaces between solids. For this application, defining solids with exact elastic properties at each point P along the principal deformation direction (or "radial direction") defined by the normal vector $N(P)$ is sufficient.

To express exact non-linear elasticity in radial directions, we let $dR(Y)$ be a small radial force and dY the resulting small radial displacement. From equation (1) they must satisfy: $k_P(Y)dY = dR(Y)$. If we express deformations by variations in the field function, it yields:

$$df(Y) = Grad(f,Y).dY = Grad(f,Y).\frac{dR(Y)}{k_P(Y)} \qquad (2)$$

As said previously, we want to use the way the field function varies inside the implicit solid to express physical properties. Let us directly model stiffness with the field's gradient:

$$\forall Y \quad Grad(f,Y) = -k_P(Y)\,N(Y) \qquad (3)$$

This choice simplifies equation (2) which yields:

$$\int_{X_0}^{X(P)} df(Y) = -\int_{X_0}^{X(P)} (N(P).dR(Y)) = -N(P).R(P) \qquad (4)$$

where the normal vector $N(P)$ remains constant during radial deformations. Let $g(P) = f(X(P)) - f(X_0)$ be the deformation field term associated at equilibrium with the radial force R. Here, the correspondence formula (4) becomes:

$$g(P) = -N(P).R(P) \qquad (5)$$

We use equation (5) to define the general correspondence between deformations and forces characterizing implicit deformable solids. Used with a field function satisfying (3), this correspondence gives exact elastic properties in radial directions, but it associates no deformation at all with forces lying in the local tangent plane. Again, this is not a problem since the formula will only be used for computing the radial component of compression forces due to collisions.

Modeling stiffness with field functions

Let S be an object defined by a single skeleton and P_S a point of this skeleton. The field function along the segment between P_S and the closest point of the surface can be expressed as a function $f(r)$ of the distance $r(P) = d(P, P_S)$. From equation (3), the local stiffness at point P satisfies: $k(P)\,N(P) = -f'(r(P))\ Grad(r,P)$.
But $Grad(r,P) = (P - P_S)/||P - P_S|| = N(P)$, so:

$$k(P) = -f'(r(P))$$

The resulting geometric representation of stiffness (the opposite of the field function's slope) facilitates the control of the simulated material. The user does not need to be a specialist in mathematical physics to easily design linear and non-linear elastic models as those of Figure 2. The field functions currently implemented are given in Appendix A.

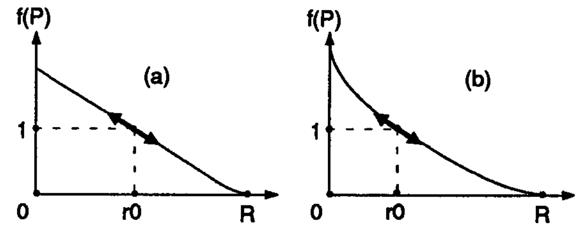


Figure 2: Examples of field functions.
(a) Linear elasticity: stiffness is constant during deformations.
(b) Non-linear elasticity: stiffness increases during compressions.

Homogeneity of the solids

When an object is generated by several skeletons, different field functions f_i can be associated with each one allowing non-homogeneous objects to be readily designed. The object behaves according to the local stiffness k_i in a zone influenced by a single skeleton. Otherwise, stiffness contributions from different skeletons blend together:

$$k(P)N(P) = -Grad(f,P) = -\sum Grad(f_i,P)$$

When "distance surfaces" are used, summing stiffness contributions in blending areas can be a problem. The stiffness may pass by a local extremum while varying between values associated with different skeletons. This problem corresponds to the bulge (or the narrowing) in shape which can appear when two fields superimpose [3]. Indeed, there is no reason why $||\sum_{j=1}^{n} Grad(f_j,P)||$ should take intermediate values between the k_i. In particular, homogeneous objects are not easy to model with distance surfaces. Giving the same field function to all the skeletons is far from sufficient.

"Convolution surfaces", for which field functions f_i are integrals of field contributions from each point of the associated skeleton, solve this problem. With this model, if the same field functions are used for several neighboring skeletons there is no bulge in shape nor in the stiffness function, so complex homogeneous objects can be designed. More generally, stiffness smoothly assumes intermediate values in areas influenced by multiple skeletons as does the field's gradient.

2.3 Animation of implicit deformable solids

Implicit deformable solids are especially suitable for a precise modeling of interactions. While penalty methods directly use the degree of interpenetration between objects to evaluate response forces, our model completely suppresses interpenetrations by introducing an intermediate "contact modeling" step between the detection and the response to collisions. The general animation algorithm is the following: At each time step,

1. Integrate the equations of motion for the rigid components of the solids by taking external forces F and torques T into account:

$$\begin{aligned}\sum F &= mA\\ \sum T &= I\dot{\Omega} + \Omega \wedge I\Omega\end{aligned}$$

where I is the matrix of inertia of a solid computed from its rest shape. A represents linear acceleration and $\dot{\Omega}$ angular acceleration.

2. Displace flexible components from their rest shapes.
3. Treat interactions between objects:
   (a) Detect interpenetrations.
   (b) Model contact by deforming each solid in order to generate contact surfaces.
   (c) Integrate reaction and friction forces. Add them to the set of external actions to be applied to the rigid components at the next time step.
4. Display the objects with their new deformed shapes.

This algorithm is used with an adaptive time step. As with penalty methods, overly deep interpenetrations generate overly large response forces (resulting, in the modeling contact phase, in excessive deformation of the objects). When this situation is detected, the system recomputes the objects positions using a smaller time interval.

The next section details the three steps of the interaction processing module and studies extensions to multiple collisions and to interactions with rigid implicit solids.

3 Interactions between Implicit Solids

3.1 Interpenetration detection

We use axis-parallel bounding boxes to quickly cull most non-intersecting cases. Afterwards, we benefit from the implicit representation of the objects, as in [10]. For each pair of solids, sample points associated with one of them are tested against the inside/outside function of the other. This is done, of course, only for the sample points located inside the second solid's bounding box. As the list of solids interacting together is the only information needed for the modeling contact step, detection is stopped for a given pair of solids as soon as an interpenetration point is found.

The method used for computing sample points at each time step is detailed in Section 4. As will be shown, the detection process can be optimized by starting detection in the neighborhood of points (if any) that most penetrated the other object during the last time step.

3.2 Modeling contact

Once detected, an interpenetration must be suppressed by deforming each object according to the set of interacting solids. Deforming objects involves generating contact surfaces as well as modeling the transverse propagation of deformations (see Figure 3). Rather than simulating local interactions inside the objects, the system directly computes deformed shapes at equilibrium using a model for damped propagation. Deformations outside a given "propagation area", an offset of the interpenetration zone, are ignored.

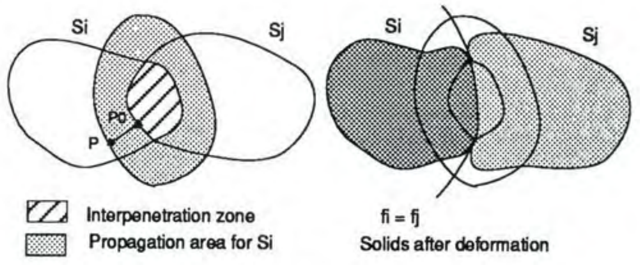


Figure 3: Modeling contact consists in applying different deformation fields in the interpenetration zone and in the "propagation area" associated with each solid (view in cross section).

Interpenetration areas: Generating contact surfaces

In addition to being a very natural model to express physical properties, the implicit surface formalism is also convenient for generating exact contact surfaces. See Figure 4.

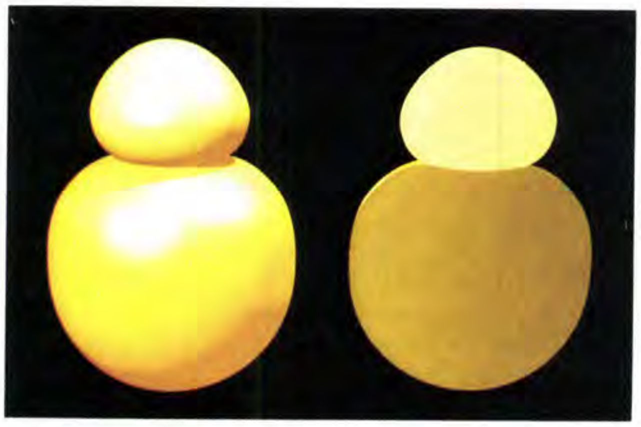


Figure 4: (left) Contact between two colliding objects.
(right) View in cross section showing the exact contact modeling.

Suppose that two objects S_i and S_j interact locally. We are looking for new terms g_{ji} and g_{ij} to add to their respective field functions f_i and f_j in the interpenetration zone (g_{ji} represents the action of object j on object i). After deformation, the objects will be defined in this area by:

$$f_i(P) + g_{ji}(P) = 1 \tag{6}$$

$$f_j(P) + g_{ij}(P) = 1 \tag{7}$$

The deformation fields g_{ji} and g_{ij} must be negative (they model local compression of the objects) and locally generate a contact surface, thus equations (6) and (7) must have common solutions. In order to give the new contact surface exactly the same border as the interpenetration area, deformation fields must satisfy $g_{ij}(P) = g_{ji}(P) = 0$ in points P where $f_i(P) = f_j(P) = 1$. Moreover, the contact surface generated must fit with the local rigidities of colliding objects. So opposite forces must be applied by the two compressed objects on each point of this surface. Adding extra skeletons to generate deformation field terms would be inconvenient; Indeed, we prefer to directly use S_j's skeleton to deform S_i and vice versa. Consequently, the deformation field terms of an interpenetration area are defined by:

$$\begin{aligned} g_{ji}(P) &= 1 - f_j(P) \\ g_{ij}(P) &= 1 - f_i(P) \end{aligned}$$

With this choice, all the properties needed are verified: deformation field terms are negative in the interpenetration zone, and generate a contact surface defined by:

$$f_i(P) = f_j(P) \tag{8}$$

Let $P \in S_i$ be a point of the contact surface, $N_i(P)$ be S_i's unit normal vector, and $N_j(P) = -N_i(P)$.
Let $R_i(P) = ||R_i(P)||N_i(P)$ be the radial force applied by S_j at P. The correspondence (5) between forces and deformations yields : $g_{ji}(P) = -R_i(P).N_i(P) = -||R_i(P)||$, so: $R_i(P) = -g_{ji}(P)N_i(P)$. From equation (8), opposite forces are then applied by the objects along the contact surface:

$$R_i(P) = (1 - f_j(P))\ N_j(P) = (1 - f_i(P))\ N_j(P) = -R_i(P)$$

Deformations in "propagation areas"

We want to optimize the contact modeling process by directly computing deformed shapes in contact positions rather than simulating local interactions inside the flexible material. Designing a purely geometric layer is justified here: only deformations along contact surfaces will be used for computing response forces. The use of geometric propagation will not affect the motion at all. Moreover, we wish to model damped material where deformations outside given "propagation areas" can be neglected. Providing the user with a set of intuitive parameters, such as the thickness of the propagation areas around interpenetration zones or the way deformations are attenuated, offers a simple and efficient control of the simulated material.

More precisely, the user controls S_i's propagation field term $p_{ji}(P)$ (due to the collision with S_j) through two additional parameters in S_i's description:

- A thickness value w_i giving the size of the offset were deformations propagate around an interpenetration zone. Deformations will be neglected outside this area.
- An "attenuation value" α_i giving the ratio between the maximal value desired for p_{ji} and the current maximal compression term in the interpenetration area.

Because of the parameter α_i, the size of the bulge due to propagation of deformations will first increase during a collision, while the solid is progressively compressed, and then decrease back to zero when the colliding objects move off.

The propagation field p_{ji} must be positive within the propagation area in order to model a local expansion of the solid, compensating for the compression due to collision. To preserve the shape's first order continuity[2] p_{ji} and its derivative must become zero at the exterior limit of the propagation area, and have the same value and gradient vector as the contact term g_{ji} in the border of the interpenetration zone. Let P be a point within the propagation area, and P_0 the closest point of S_j in S_j's gradient direction (see Figure 3). To satisfy the conditions just listed, we define p_{ji} along the line (P_0, P) by:

$$p_{ji}(P) = a_{k,a0,w_i}(d(P, P_0))$$

where $k = ||Grad(f_j, P_0)||$, a_0 is the maximal propagation value equal to α_i times the maximal compression field value, and $a_{k,a0,w}(x)$ is the piecewise polynomial function shown in Figure 5. An exact formula is given in Appendix B.

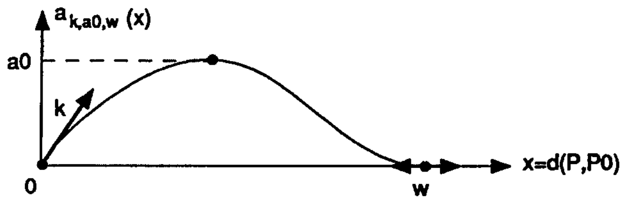


Figure 5: Attenuation function defining the propagation field.

With this choice, all the conditions on p_{ji} can be verified. Let us prove that $Grad(p_{ji}, P_0) = Grad(g_{ji}, P_0)$.

$$Grad(p_{ji}, P) = a'_{k,a0,w_i}(d(P, P_0))\ N_0$$

With the value of $a_{k,a0,w_i}$'s derivative in zero, we obtain:

$$Grad(p_{j,i}, P_0) = -Grad(f_j, P_0) = Grad(g_{ji}, P_0)$$

Expansion of the objects in propagation areas must not produce new interpenetrations. To best avoid this situation, we insure that each deformed object does not cross the median surfaces of equation $f_i(P) = f_j(P)$ (see Figure 3). In other words, $p_{j,i}(P)$ must be less than or equal to $1 - f_j(P)$ throughout the propagation area. If the problem does occur, the system truncates the propagation term and issues a warning that a smaller value should be chosen for α_i.

3.3 Computation of response forces

Radial reaction forces

The reaction forces directed along normal vectors are given by the correspondence (5) between forces and deformations. They are are numerically integrated along contact surfaces (this process is detailed in Section 4). Because of our choice for the contact surface, the principle that opposite reactions occur on two colliding objects is verified.

Friction and damping forces

To model both tangential friction in contact areas and damping due to the progressive compression of the solids, we include a friction coefficient λ_i in the description of each object. When a collision occurs, the friction and damping force F_i at a point P of the contact surface between S_i and S_j is expressed by:

$$F_i(P) = \lambda_i \lambda_j (V_j(P) - V_i(P)) \tag{9}$$

[2]The method would be easy to extend to higher order continuity by considering constraints over higher order derivatives, and by using more complex attenuation functions.

where $V_i(P)$ (respectively $V_j(P)$) is the speed of P, a point on the surface of the solid S_i (respectively S_j). Like radial reaction forces, friction forces are numerically integrated along contact surfaces.

Figure 6 shows the action of response forces during a few steps of an animation.

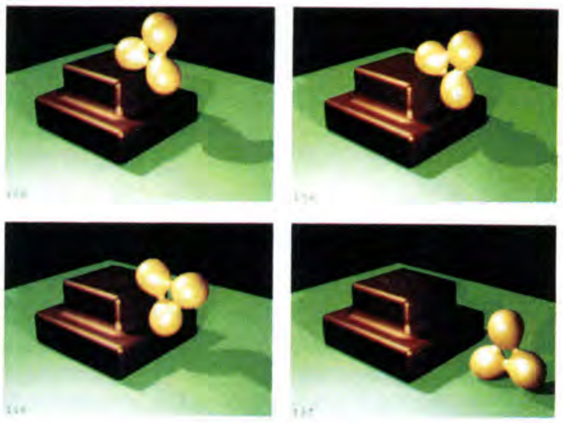


Figure 6: Flexible clover falling on a quite rigid staircase.

From collisions to lasting contacts

The deformed shapes generated during the contact modeling step can be conveniently used for lasting contacts and equilibrium states, because opposite forces are applied to each side of a contact surface. In Equation (9), F_i's tangential component represents friction due to the different tangential speeds of the solids at a contact point, while the normal component models the loss of energy due to the progressive deformation of the solids. The energy consumed over time enables colliding objects to settle into lasting contact situations, and then into resting stable states without unwanted oscillations. Figure 7 is an example of equilibrium state between four solids.

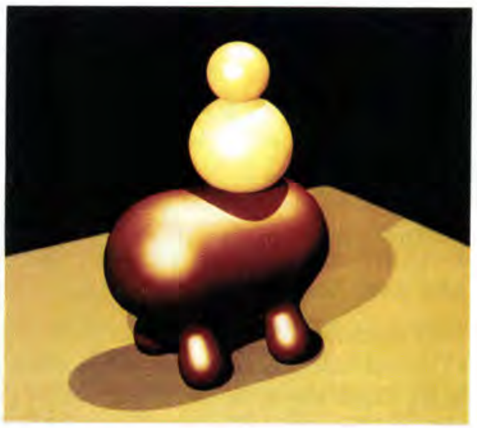


Figure 7: An equilibrium state between a rigid floor, a flexible vaulting horse, and two soft balls.

3.4 Multiple interactions

An important benefit of our model is that, in multiple interaction situations such as in Figure 7, the resulting shapes and reaction forces are *completely independent* of the order in which objects, or pairs of objects, are considered.

When an object interacts with several others, its compression field term (which produces the contact forces) is defined as a sum of terms due to the different collisions. No propagation term must be added in an interpenetration zone with another object, so, in practice, we always use a procedural method to compute field values. To evaluate the field generated by a deformed object S_0 at a given point P:

1. Compute the initial field value $f_0(P)$.
2. For each object S_i interacting with S_0, if P lies inside S_i, add the contact deformation term $1 - f_i(P)$.
3. If P was not lying inside any of the S_i, compute and sum all non-zero propagation terms at P. Truncate this sum if needed (as explained at the end of Section 3.2) before adding it to the field value.

If an intersection area is detected between more than two solids as in Figure 8, several negative compression terms are simultaneously added in this area. This leaves a small space between the solids, whose shapes remain C^1 continuous (generating multiple contact points would produce singularities).

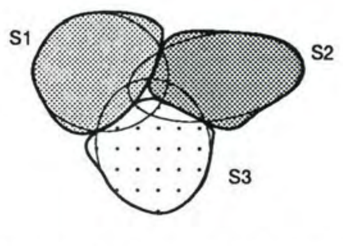


Figure 8: Deformation of 3 intersecting solids (cross sections).

3.5 Interactions with rigid implicit solids

Another important issue for our model is its ability to simulate interactions between flexible and rigid objects[3], like the vaulting horse and the floor in Figure 7.

Suppose an interpenetration has been detected between a rigid solid S_j and a flexible object S_i. The deformation field term applied to S_i in the contact area must make S_i exactly fit S_j's shape; i.e., the solutions of $f_i(P)+g_{ji}(P)=1$ must be points satisfying $f_j(P)=1$. Moreover, the deformation field term must be negative in the interpenetration zone given by $f_i(P) \geq 1, f_i(P) \geq 1$. Then, we define $g_{ji}(P)$ by:

$$g_{ji}(P) = (1 - f_j(P)) + (1 - f_i(P))$$

The usual formula is used for the attenuation function in S_i's propagation area. Simply, S_j's gradient vector (used to

[3]Analytical solutions such as those developed in [1] should be used for interactions between pairs of rigid objects.

define the slope of the attenuation function) is now replaced by $-(Grad(f_i + f_j, P_0))$ so that the bulge will exactly fit S_j's normal vectors at the border of the contact surface. When this is done, response forces corresponding to S_i's deformation are integrated along the contact surface, and opposite forces are applied to the rigid solid S_j according to the principle that opposite reactions occur on two colliding objects.

4 Implementation

Our modeling and animation system for implicit deformable solids is implemented in C++ on an SGI Indigo workstation. The current implementation uses distance surfaces which provide us with analytical expressions of normal vectors.

4.1 Optimizing the animation process

One of the main problems raised by implicit isosurfaces is the search for efficient ways to discretize objects. The animation process uses discretizations three times by animation step: for collision detection, for integrating response forces, and for displaying objects. Despite recent improvements in adaptive octree techniques, spatial partitioning polygonizations remain quite expensive. Using this type of algorithm at each time step would prevent any interactive computation and display of the animation.

Fortunately, the solids only deform locally during animations and return to their rest shapes. Their topology never changes (otherwise, our hybrid model with its invariant matrix of inertia would be invalid). Consequently, the objects need not be completely re-sampled at each time step.

Before an animation is calculated, sample points and the associated normal vectors are precomputed form the object's rest shape, and stored relative to the local coordinate system. Then, at each animation step:

- The sample points are positioned according to the current position and orientation of the solid's rigid component.
- They are used to detect collisions. To benefit from temporal coherence, tests for interpenetration are first performed in the neighborhood of points P_j which penetrated most deeply into the other object at the previous time step[4].
- Finally, the sample points in deformed areas are recomputed, before display, by using a linear search algorithm along the undeformed normal direction. The use of this direction insures that the points will come back to their initial positions after any deformation.

  To improve in efficiency, response forces in contact areas are integrated during this process. If a point is located in an interpenetration zone, the field function computes the deformation term, which is equal to the local reaction force. As soon as a point of the contact surface is found, this force (plus the friction force term), multiplied by the area of an elementary surface ds, is added to the sum of external actions applied to the object. In the current implementation, ds is approximated by an average value computed from the size of the discretization voxels.

[4]We use a continuation method for sampling, so starting detection a few points before P_j in the list of sample points is sufficient.

4.2 Rendering

The implicit formalism provides us with exact high level descriptions of deformed solids, even if only a few sample points are used during the computations. During animations, we directly save the parameters needed to compute the deformed field functions defining the objects (including stiffness, scope of influence, attenuation parameters, and the current list of colliding objects). To give an idea of required disc space, the file describing the implicit objects of Figure 7 take less than 1 Kbyte, while the storage of the associated sample points with their normal vectors and the list of triangles takes more than 1200 Kbyte.

Once the objects are stored, any method could be used for rendering, including computing polygonizations with an arbitrary precision. We currently use direct ray-tracing on implicit surfaces, implemented as an extension to the public domain renderer *Rayshade* (by C. Kolb). If a ray intersects an implicit solid's bounding box, we first look for a seed point along the ray which is located inside the solid. If we find one, an intersection point is computed by binary search. Testing if a point is inside or outside is done by evaluating the solid's potential field. Normal vectors at the intersection points are analytically computed from the field gradient.

5 Conclusion

This paper presents a novel way to model deformable solids. The implicit formalism provides a compact and unified formulation for both geometric and physical properties, and enables to keep a continuous high level representation of the objects. The model offers simple and quite general control of the simulated material. One can experiment with field function curves to adjust stiffness variations when objects are compressed, or with different ways to propagate deformations due to collisions. All the parameters are easy to understand, even for a non-specialist.

Well adapted to local deformations, the system is especially designed for a precise modeling of interactions. It generates exact contact surfaces between solids which facilitates a precise evaluation of reaction forces. The model applies to sudden collisions, lasting contacts, and equilibrium situations. It gives an elegant solution to the multiple collision problem, producing new deformed shapes and response forces which are independent of the order in which collisions are detected. The model can be generalized to treat interactions between flexible and rigid objects.

During animations, discretized representations of the objects are displayed at interactive rates, while a compact storage of their implicit description is performed. This description, which still defines curved contact surfaces, is used for producing subsequent, high-quality images.

Future work

Implementing convolution surfaces [3] would facilitate the design of complex objects composed of homogeneous materials. To improve generality, these surfaces should be extended to non-exponential potential fields.

At present, deformed shapes only depend on the set of external forces currently applied to the solids, so deformations disappear as soon as there is no longer contact between objects. Modeling visco-elastic or elasto-plastic behaviors in addition to pure elasticity would be a good extension. Moreover, some objects of the real world conserve their volume during deformations while others are partially compressible. Modeling the propagation of deformations according to a compressibility parameter would provide easier control.

The implicit deformable model opens new directions for future research, particularly within the area of human simulation. Modeling complex objects such as deformable flesh covering rigid skeletons could not be done with previous global deformation techniques (the skeleton must not be deformed, nor the flesh on the opposite side of the skeleton). Nodal approaches can be used, but they demand complicated databases [6] and involve an expensive numerical simulation of deformations propagating in damped material. Our ability to model local deformations while preserving a compact continuous representation of objects, even when they are non-homogeneous, would be helpful. Moreover, the precise contact modeling presented here should allow human models to interact with the simulated world.

Acknowledgements

Many thanks to Jean-Dominique Gascuel for his constant support during this research and for implementing direct ray-tracing on implicit surfaces. Thanks to Philippe Limantour, Christophe Vedel, Frédéric Asensio and Julien Signes for interesting early discussions, and to François Sillion, Alain Chesnais, Dave Forsey, Phil Brock and Jules Bloomenthal for re-reading this paper.

References

[1] David Baraff. Dynamic simulation of non-penetrating rigid bodies. *PHD Thesis*, Cornell University, May 1992.

[2] David Baraff and Andrew Witkin. Dynamic simulation of non-penetrating flexible bodies. *Computer Graphics*, 26(2):303–308, July 1992. Proceedings of SIGGRAPH'92 (Chicago, Illinois, July 1992).

[3] Jules Bloomenthal and Ken Shoemake. Convolution surfaces. *Computer Graphics*, 25(4):251–256, July 1991. Proceedings of SIGGRAPH'91 (Las Vegas, Nevada, July 1991).

[4] Jules Bloomenthal and Brian Wyvill. Interactive techniques for implicit modeling. *Computer Graphics*, 24(2):109–116, March 1990.

[5] Marie-Paule Gascuel, Anne Verroust, and Claude Puech. A modeling system for complex deformable bodies suited to animation and collision processing. *Journal of Visualization and Computer Animation*, 2(3), August 1991. A shorter version of this paper appeared in *Graphics Interface'91*.

[6] Jean-Paul Gourret, Nadia Magnenat Thalmann, and Daniel Thalmann. Simulation of object and human skin deformations in a grasping task. *Computer Graphics*, 23(3):21–29, July 1989. Proceedings of SIGGRAPH'89 (Boston, MA, July 1989).

[7] Annie Luciani, Stéphane Jimenez, Olivier Raoult, Claude Cadoz, and Jean-Loup Florens. An unified view of multitude behaviour, flexibility, plasticity, and fractures: balls, bubbles and agglomerates. In *IFIP WG 5.10 Working Conference*, Tokyo, Japan, April 1991.

[8] Gavin Miller. The motion dynamics of snakes and worms. *Computer Graphics*, 22(4):169–177, August 1988. Proceedings of SIGGRAPH'88 (Atlanta, August 1988).

[9] Matthew Moore and Jane Wilhelms. Collision detection and response for computer animation. *Computer Graphics*, 22(4):289–298, August 1988. Proceedings of SIGGRAPH'88 (Atlanta, August 1988).

[10] Alex Pentland and John Williams. Good vibrations: Modal dynamics for graphics and animation. *Computer Graphics*, 23(3):215–222, July 1989. Proceedings of SIGGRAPH'89 (Boston, MA, July 1989).

[11] Demetri Terzopoulos and Dimitri Metaxas. Dynamic 3-D models with local and global deformations : deformable super quadrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-13(7):703–714, July 1991.

[12] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. *Computer Graphics*, 21(4):205–214, July 1987. Proceedings of SIGGRAPH'87 (Anaheim, California, July 1987).

[13] Demetri Terzopoulos and Andrew Witkin. Physically based model with rigid and deformable components. *IEEE Computer Graphics and Applications*, pages 41–51, December 1988.

[14] Andrew Witkin and William Welch. Fast animation and control for non-rigid structures. *Computer Graphics*, 24(4):243–252, August 1990. Proceedings of SIGGRAPH'90 (Dallas, Texas, August 1990).

Appendix A. Equation for the field functions

Field functions currently implemented are parameterized by a scope of influence R, a thickness value r_0, and a stiffness value k:

$$\begin{aligned} f_i(P) &= ar^2 + br + c && \text{if } r \in [0, r_0] \\ f_i(P) &= (r-R)^2(dr+e) && \text{if } r \in [r_0, R] \\ f_i(P) &= 0 && \text{elsewhere} \end{aligned}$$

$d = -(k(r_0 - R)+2)/(r_0 - R)^3$, $e = (kr_0(r_0 - R)+3r_0 - R)/(r_0 - R)^3$.

If linear elasticity is chosen, $a = 0$, $b = -k$ and $c = kr_0 + 1$. If non-linear elasticity is selected, we use $a = k/(2r_0)$, $b = -2k$, $c = 3kr_0/2 + 1$ (of course any other stiffness variation inside the object would be easy to implement).

Appendix B. Equation for the attenuation function

The equation we use for the attenuation function is:

$$\begin{aligned} a_{k,a0,w}(r) &= cr^3 + dr^2 + kr && \text{if } r \in [0, w/2] \\ a_{k,a0,w}(r) &= \frac{4a0 + (x-w)^2(4x-w)}{w^3} && \text{if } r \in [w/2, w] \end{aligned}$$

Where $c = 4(wk - 4a0)/w^3$ and $d = 4(3a0 - wk)/w^2$.

Interval Methods for Multi-Point Collisions between Time-Dependent Curved Surfaces

John M. Snyder, Adam R. Woodbury,
Kurt Fleischer, Bena Currin, Alan H. Barr
California Institute of Technology
Pasadena, CA 91125

Abstract

We present an efficient and robust algorithm for finding points of collision between time-dependent parametric and implicit surfaces. The algorithm detects simultaneous collisions at multiple points of contact. When the regions of contact form curves or surfaces, it returns a finite set of points uniformly distributed over each contact region.

Collisions can be computed for a very general class of surfaces: those for which inclusion functions can be constructed. Included in this set are the familiar kinds of surfaces and time behaviors encountered in computer graphics.

We use a new interval approach for constrained minimization to detect collisions, and a tangency condition to reduce the dimensionality of the search space. These approaches make interval methods practical for multi-point collisions between complex surfaces. An interval Newton method based on the solution of the interval linear equation is used to speed convergence to the collision time and location. This method is more efficient than the Krawczyk–Moore iteration used previously in computer graphics.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling; G.4 [Mathematical Software]: Reliability and Robustness

General Terms: collision detection, parametric surface, constrained minimization, interval analysis

Additional Key Words: inclusion function, interval Newton method, interval linear equation

1 Introduction

Detecting geometric collisions between curved, time-dependent (moving and deforming) objects is an important and difficult problem in computer graphics. This paper discusses a practical and robust algorithm for detecting collisions between objects represented as parametric or implicit surfaces. We ignore the problem of computing the physical response to collisions; much of this topic is treated in other work [BARA90,META92]. Instead, we concentrate on the purely geometric problem of computing a solution set of points where a set of time-dependent surfaces first contact (Figure 1).

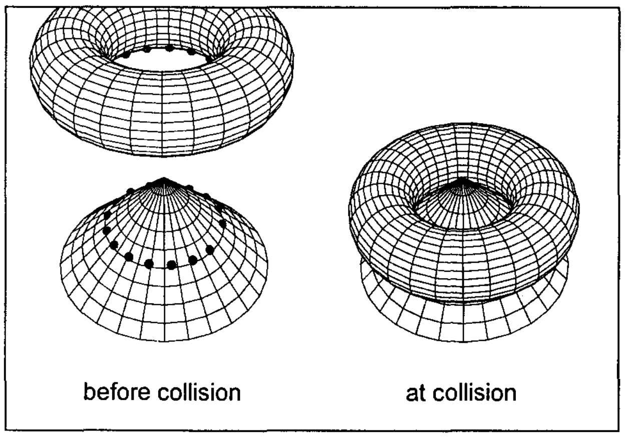


Figure 1: Problem Statement: Given a collection of time-dependent curved surfaces, find a set of collision points representing the contact regions. In this example, the dots show the points detected by the collision algorithm when a torus moves down over a cone, contacting it in a circle.

Previous work on geometric collision detection is fairly extensive, both in computer graphics and in other fields such as CAD/CAM and robotics. Detection of collisions between polyhedral objects was studied in [MOOR88]. Baraff [BARA90] presented a method of computing collisions between parametric or implicit surfaces by computing extremal points using non-linear equation solvers. Sclaroff and Pentland [SCLA91] present a method for detecting collisions between implicit surfaces by "plugging" vertices of a polyhedral approximation of one surface into the inside-outside function of the other. Von Herzen, et. al., [VONH90] presented an algorithm for detecting collisions of parametric surfaces using Lipschitz bounds. Duff [DUFF92] used interval methods to compute collisions between boolean combinations of implicit surfaces.

To make collision detection practical, much of the previous work traded off accuracy and robustness for efficiency, or limited the kinds of shapes that could be handled. Polyhedral methods such as in [MOOR88], although fairly efficient, are not well suited to surfaces that deform in time. Exploiting coherence for rolling or sliding contact of polyhedral objects is difficult, and use of a fixed sampling mesh can cause severe approximation errors. Polyhedral methods also require many numerically difficult special cases which led [MOOR88] and [SCLA91] to neglect cases where "tunneling" may occur either between polygon edges or between small implicit surfaces passing entirely through a large polygon.

Baraff [BARA90] chose to limit objects to the union of con-

```
take one or more steps in the ODE solver
compute collisions in the resulting time interval
if a collision occurs (at time $t^*$) in the interval
      compute a collision response
      reset ODE solver to $t = t^*$
endif
```

Figure 2: Computational Model for Collision Detection and Response

vex polyhedra and strictly convex closed surfaces. This restriction simplified his collision detection algorithm and allowed tracking of single contact points between curved objects. He did not treat non-convex surfaces (such as saddle shapes) and manifolds with boundary (such as half a sphere). We solve the problem for a more general class of surfaces with many points of contact, as shown in Figure 1.

As noted in [VONH90], methods which depend solely on point-wise evaluations, including the above methods, cannot guarantee accurate collision detection. To solve this problem, Von Herzen bounded the output of functions over a region using a Lipschitz bound. Duff [DUFF92] used interval analysis to produce tighter bounds than Von Herzen's Lipschitz bound. Both of these methods used binary subdivision to search for collisions; we speed up the approach significantly by combining binary subdivision with an interval Newton method.

The technique we describe offers several fundamental improvements over previous techniques:

1. The most novel aspect of our technique is the ability to detect simultaneous collisions (multiple contacts at the same time), *even when the collisions occur at a higher dimensional manifold of contact, rather than at a set of isolated points.* In this case, the algorithm samples the region of contact with a finite, uniformly-distributed set of points. The spatial sampling density is a parameter to the algorithm. To our knowledge, no previous algorithm handles this situation.
2. Our technique works for both rigid and deforming objects, and for implicit or parametric objects.
3. Our technique is practical for computer graphics applications, and has been used in animations involving hundreds of objects.
4. Our technique includes a method (tangency constraints) to reduce the dimensionality of the space of possible solution points, as shown in Figure 3, dramatically speeding up the method. The tangency constraints also provide a square system of equations for the interval Newton method, helping us detect isolated point collisions.
5. Our technique uses a test for uniqueness of roots of a system of equations in a region. This test can be verified in many cases, allowing the algorithm to terminate without further subdivision around collision points.
6. Our technique can be used both to compute collisions between formerly disjoint bodies which come into contact, or to compute additional points of contact between bodies as they roll or slide over each other (see Section 1.1).

1.1 Fitting Collision Detection into a Larger System

Figure 2 shows how collision detection fits into a larger program for computing physical simulations of dynamic systems. The system is composed of three parts: the ODE (ordinary differential equation) solver module, the collision detection module, and the collision response computation module. The ODE solver computes the motions of objects over time, using equations governing the dynamic behavior of bodies, and produces a functional representation of the motion.[1] Motion is computed without considering collisions, so that the results are only valid until the next collision occurs. The collision detection module takes the functional representation produced by the ODE solver and computes when and where the first collision occurs in the given time interval. If a collision occurs, a collision response is computed, which may discontinuously change the state of the system of bodies. The ODE solver continues forward in time from this computed collision time, discarding any state after it.

Two modes of operation are required in collision detection:

1. compute any collisions for bodies that are initially not in contact
2. compute *additional* collisions for bodies that are already in continuous (rolling or sliding) contact

The algorithm described in this paper handles both situations. For greatest efficiency and modularity, we advocate handling coherence in the ODE solver. By coherence, we mean the tracking of contact points between bodies rolling or sliding over each other. In these situations, collision detection is required only to compute new points of contact not already tracked by the ODE solver (mode 2 above). The solver must therefore inform the collision detection module of the motion of the contact points it is tracking, so that these points may be excluded from consideration (see Eq. 7). The collision detection module must also compute the initial points of contact when the simulation is begun or when continuous contact begins between bodies (mode 1 above).

1.2 Overview

The mathematics of the collision detection problem is treated in Section 2. Sections 3, 4, and 5 discuss the constrained minimization algorithm, an interval Newton enhancement, and termination criteria, respectively. Section 6 presents a simple culling test which discards non-colliding surface pairs and tightens a bound on the collision time. The full collision algorithm, combining constrained minimization, the culling test, and other tools from computational geometry, is presented in Section 7. Our technique, like all interval methods, requires inclusion functions, whose construction is summarized in Section 8. Finally, results and conclusions are described in Sections 9 and 10. Appendix A extends our approach to surfaces that are piecewise smooth by adding conditions for face, edge, and vertex interactions (see Figure 11). Appendix B describes the construction of inclusion functions for Chebyshev polynomials.

2 The Collision Problem

The equations that specify that two surfaces collide may be divided into two parts: a *contact* constraint, that specifies that the two surfaces intersect, and a *tangency* constraint, that specifies that the two surfaces are tangent at their point of intersection. The tangency constraint reduces the dimensionality of the space of possible collision points, as shown in Figure 3. It also allows faster convergence (using interval Newton, which we will describe in Section 4)

[1] In our rigid body simulations, the solver produces a time-varying quaternion and translation vector. Each component of the quaternion and vector is represented using univariate Chebyshev polynomials.

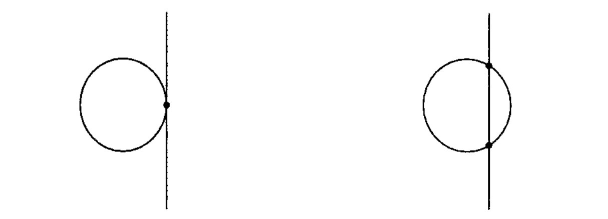


Figure 3: Reducing the Dimensionality of the Space of Collision Points Using the Tangency Condition: The intersection of two bodies (like a sphere moving to the right with a stationary plane) typically forms a whole 2D manifold of contact through time. With the tangency constraint, the solution space is often reduced to one or a few points by eliminating cases like that shown on the right. Reducing the solution space to an isolated space-time point is one of the ideas that makes this method practical.

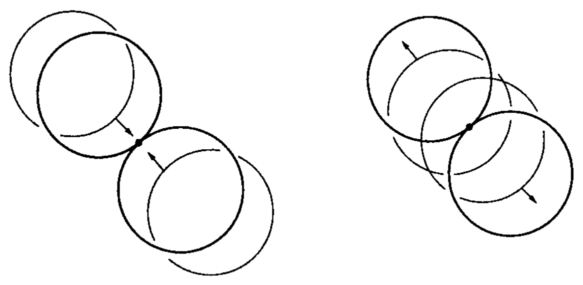


Figure 4: Incoming and Outgoing Collisions: The unbroken circles represent bodies later in time. A dot represents the collision point; the arrows represent the direction of movement.

and robust testing of isolated collisions (using an interval solution uniqueness test described in Section 5).

We also distinguish between *incoming* collisions, in which the surfaces collide by moving closer to each other, and *outgoing* collisions, in which the surfaces are interpenetrating and become tangent as they move apart. These situations are compared in Figure 4. The distinction is necessary in the simulation of dynamic systems where each surface encloses a solid. Eliminating outgoing collisions allows the simulator to ignore collisions which were previously detected; i.e., collisions between surfaces already in contact which are moving away as a *response* to the collision.

2.1 Parametric Surfaces

Let two deforming parametric surfaces be represented by the twice-differentiable mappings $S_1(u_1, v_1, t)$ and $S_2(u_2, v_2, t)$, where $S_i\colon \mathbf{R}^3 \to \mathbf{R}^3$. At a particular instant of time, each of the surfaces is formed by the image of S_i over a rectangle in (u_i, v_i) space.[2] In this section, we consider the case of collisions between solids each bounded by a single, smooth, closed parametric surface. Appendix A generalizes the discussion to parametric surfaces which are only piecewise smooth.

Contact Constraint The contact constraint merely states that the two surfaces intersect (i.e., the vector difference of the two surfaces is the zero vector):

$$S_1(u_1, v_1, t) - S_2(u_2, v_2, t) = 0. \tag{1}$$

Tangency Constraint The tangency constraint implies that the instantaneous normal vectors on the two surfaces at their point of contact are anti-parallel. Stated another way, the (u, v) tangent vectors on one surface must be perpendicular to the instantaneous normal vector on the other surface. We thus have the following system of two equations[3]

$$\begin{pmatrix} \frac{\partial S_1}{\partial u_1}(u_1, v_1, t) \cdot N_2(u_2, v_2, t) \\ \frac{\partial S_1}{\partial v_1}(u_1, v_1, t) \cdot N_2(u_2, v_2, t) \end{pmatrix} = 0 \tag{2}$$

where N_1 and N_2 are the outward normal vectors to the surfaces S_1 and S_2, respectively, given by

$$N_i(u_i, v_i, t) \equiv \frac{\partial S_i}{\partial u_i}(u_i, v_i, t) \times \frac{\partial S_i}{\partial v_i}(u_i, v_i, t) \quad \text{for } i = 1, 2.$$

The algorithms that follow here assume that N_1 and N_2 are nowhere 0; that is, surfaces have a nonvanishing normal vector everywhere and for all relevant time.[4] The whole collision equality constraint is given by a nonlinear system of 5 equations in 5 variables, three from Eq. 1 and two from Eq. 2.

Incoming Constraint The incoming collision condition states that the relative velocity of the collision point must face the same way as the surface normal (the two vectors must form an acute angle),[5] and the two normals must face in opposite directions (forming an obtuse angle). This condition yields two inequality constraints:

$$\begin{aligned} &\left(\frac{\partial S_1}{\partial t}(u_1, v_1, t) - \frac{\partial S_2}{\partial t}(u_2, v_2, t)\right) \cdot N_1(u_1, v_1, t) \geq 0 \\ &\text{and} \quad -N_1(u_1, v_1, t) \cdot N_2(u_2, v_2, t) \geq 0. \end{aligned} \tag{3}$$

2.1.1 Example: Rigid Parametric Surfaces

The above constraints may be applied to the special case of rigid parametric surfaces. In this case, we have two time-independent surfaces $s_1(u_1, v_1)$ and $s_2(u_1, v_1)$. The time-varying version of these surfaces is given by

$$S_i(u_i, v_i, t) \equiv R_i(t) s_i(u_1, v_1) + T_i(t) \quad \text{for } i = 1, 2$$

where $R_i(t)$ is a time-varying rotation matrix and $T_i(t)$ is a time-varying translation vector, specifying the trajectory of surface i's coordinate origin.

Contact Constraint The contact constraint may be expressed as

$$R_1(t) s_1(u_1, v_1) + T_1(t) - R_2(t) s_2(u_2, v_2) - T_2(t) = 0.$$

[2]Using a rectangular domain for parametric surfaces does not limit the kinds of surfaces that can be collided. Parametric surfaces defined on non-rectangular domains can be handled by mapping a rectangle into the required non-rectangular domain before mapping onto the surface [SNYD92b].

[3]A similar, though functionally dependent, constraint may be derived by switching S_1 and S_2.

[4]If the calculated normal vector becomes zero, such as at the poles of a parametric sphere, the tangency constraint becomes trivially true. The algorithm will therefore rely on the contact constraint to detect a collision in this case.

[5]We assume here that the surfaces are parameterized so that the normals N_1 and N_2 face outward.

Tangency Constraint Let $n_1(u_1, v_1)$ and $n_2(u_2, v_2)$ be the time-independent normals of the surfaces s_1 and s_2, given by

$$n_i(u_i, v_i) \equiv \frac{\partial s_i}{\partial u_i}(u_i, v_i) \times \frac{\partial s_i}{\partial v_i}(u_i, v_i).$$

The time-varying surface normals can therefore be expressed as

$$N_i(u_i, v_i, t) \equiv R_i(t)\, n_i(u_i, v_i)$$

since $R_i(t)$ is a rotation matrix. The tangency constraint is then given by

$$\begin{pmatrix} (R_1(t)\,\frac{\partial s_1}{\partial u_1}(u_1, v_1)) \cdot N_2(u_2, v_2, t) \\ (R_1(t)\,\frac{\partial s_1}{\partial u_2}(u_1, v_1)) \cdot N_2(u_2, v_2, t) \end{pmatrix} = 0.$$

Incoming Constraint The incoming constraint is given by

$$\left[\dot{R}_1(t) s_1(u_1, v_1) + \dot{T}_1(t) - \dot{R}_2(t) s_2(u_2, v_2) - \dot{T}_2(t)\right] \cdot N_1(u_1, v_1, t) \geq 0$$

$$\text{and} \quad - N_1(u_1, v_1, t) \cdot N_2(u_2, v_2, t) \geq 0$$

where $\dot{R}_i$ and $\dot{T}_i$ are the time derivatives of the rotation matrix and translation vector of the two surfaces.

2.2 Implicit Surfaces

Let two time-varying implicit surfaces be represented using the scalar functions $F_1(x, y, z, t)$ and $F_2(x, y, z, t)$. Points on each surface are defined as the zero-sets of these functions.

Contact Constraint The contact constraint is the system of two equations

$$\begin{pmatrix} F_1(x, y, z, t) \\ F_2(x, y, z, t) \end{pmatrix} = 0. \tag{4}$$

Tangency Constraint Let the function $\nabla F_i(x, y, z, t)$ be the spatial gradient of the implicit functions (i.e., with respect to x, y, and z). The tangency constraint is then given by

$$\nabla F_1(x, y, z, t) \times \nabla F_2(x, y, z, t) = 0. \tag{5}$$

This constraint, although a system of three equations, contains only two functionally dependent equations. The entire collision equality constraint for implicit surfaces is thus given by a system of five (four functionally independent) equations in the four variables x, y, z, and t (Eqs. 4 and 5).[6]

Incoming Constraint The incoming constraint is given by

$$-\frac{\partial F_1}{\partial t}(x, y, z, t)\, \|\nabla F_2(x, y, z, t)\| - \frac{\partial F_2}{\partial t}(x, y, z, t)\, \|\nabla F_1(x, y, z, t)\| \geq 0 \tag{6}$$

$$\text{and} \quad - \nabla F_1(x, y, z, t) \cdot \nabla F_2(x, y, z, t) \geq 0.$$

[6]We note that detecting collisions between implicit and parametric surfaces is a simpler problem than colliding pairs of parametric or implicit surfaces. By substituting the output of the parametric surface as the input (x, y, z) of the implicit surface, a system in 3 variables, (u, v, t), results, where u and v are the parametric surface coordinates.

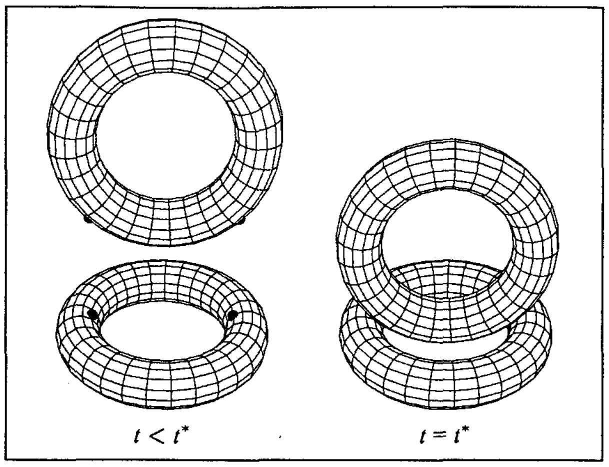


Figure 5: Simultaneous Collisions — The tori collide at two isolated points.

We also note that CSG operations on implicit surfaces, as in [DUFF92], can be handled very efficiently using our techniques. Assume the surface F_1 is represented as the boolean subtraction of two simple implicit surfaces $F_a(x, y, z, t)$ and $F_b(x, y, z, t)$. The first equation in the contact constraint (Eq. 4) then becomes

$$(F_a = 0 \text{ and } F_b \geq 0) \text{ or } (F_b = 0 \text{ and } F_a \leq 0)$$

assuming implicit surface functions are positive outside the surface they represent. Similar restricted equality constraints can be derived for the tangency constraint.

Constraints for rigid motion of implicit surfaces are easily derived by applying the above general equations to the rigidly moving implicit surface

$$F(x, y, z, t) \equiv f(w) \text{ where } w \equiv R^T(t)((x, y, z)^T - T(t))$$

where $f: \mathbf{R}^3 \to \mathbf{R}$ is the implicit equation of the time-independent surface, $R^T(t)$ is the transpose of the time-varying rotation matrix, and $T(t)$ is the time-varying translation vector.

2.3 Collision As a Constrained Minimization Problem

The final collision constraint may be described as an equality constraint involving a function C (for the contact and tangency constraints) and a logical composition of inequality constraints involving a function D (for the incoming constraint). For collisions between parametric surfaces, C and D are vector functions of (u_1, v_1, u_2, v_2, t) (equations 1–3); for implicit surfaces they are functions of (x, y, z, t) (equations 4–6). We are interested only in the minimum t collision, since our representation for the time behavior of the surfaces may be invalid after this time.

The desired collision time for parametric surfaces, t^*, can therefore be expressed using the constrained minimization problem

$$\underset{(u_1, v_1, u_2, v_2, t) \in X_0}{\text{minimum}} \left\{ t \;\middle|\; \begin{array}{ll} C(u_1, v_1, u_2, v_2, t) = 0 & \text{and} \\ D(u_1, v_1, u_2, v_2, t) \geq 0 & \end{array} \right\}.$$

A similar statement results for detection of collisions between implicit surfaces. We would like to compute t^* or detect that the constraint is satisfied nowhere in the parameter space X_0.

We also need the location of the collision and the surface normal vectors there. There may be multiple points of contact at the time of collision, which we call *simultaneous* collisions, as shown in Figure 5. The points of contact may be a finite number of *isolated points* as in Figure 5, or they may form a curve or surface, called the *contact manifold*. For example, if the falling torus from Figure 5 were in the same orientation as the stationary one at the bottom, the collision points would form a circle.

To detect simultaneous collisions, we need to detect minimum t solutions which are simultaneous or simultaneous within some tolerance ϵ. Mathematically, we require the set of *collision points* $(u_1^*, v_1^*, u_2^*, v_2^*)$ such that

$$C(u_1^*, v_1^*, u_2^*, v_2^*, t^*) = 0 \text{ and } D(u_1^*, v_1^*, u_2^*, v_2^*, t^*) \geq 0.$$

For a given collision point, the location of the collision, p^*, is

$$p^* \equiv S_1(u_1^*, v_1^*, t^*) \equiv S_2(u_2^*, v_2^*, t^*).$$

The normal vectors at the collision may be defined similarly by evaluating N_1 and N_2 at the collision point.

To compute a collision among a set of N time-varying parametric surfaces $S_i(u_i, v_i, t)$, we first use simple culling procedures to exclude pairs of surfaces which can't collide. Section 3.2 will discuss a method of solving these sets of constrained minimization problems which can compute simultaneous collisions and which does not spend undue computation on collisions which occur after t^*. It returns the collision points when these occur at a finite set of isolated points, or a finite subset of the collision points uniformly distributed over the contact manifold.

3 Interval Tools for Computing Collisions

We now turn to a discussion of the interval tools necessary to solve the sets of constrained minimization problems that arise in collisions.

3.1 Review of Interval Analysis

An *interval*, $A = [a, b]$, is a closed subset of $\mathbf{R}$ defined as

$$[a, b] \equiv \{x \mid a \leq x \leq b,\ x, a, b \in \mathbf{R}\}.$$

The lower and upper bounds of an interval are written as

$$\begin{aligned} \mathsf{lb}[a, b] &\equiv a \\ \mathsf{ub}[a, b] &\equiv b. \end{aligned}$$

A *vector-valued interval of dimension n*, $A = (A_1, A_2, \ldots, A_n)$, is a subset of $\mathbf{R}^n$ defined as

$$A \equiv \{x \mid x_i \in A_i, i = 1, 2, \ldots, n\}$$

where each A_i is an interval. An interval A_i that is a component of a vector-valued interval is called a *coordinate interval of A*.

The *width* of an interval, written $\mathsf{w}([a, b])$, is defined by

$$\mathsf{w}([a, b]) \equiv b - a.$$

The *midpoint* of an interval, written $\mathsf{mid}([a, b])$, is defined by

$$\mathsf{mid}([a, b]) \equiv \frac{a + b}{2}.$$

Similarly, the width and midpoint of a vector-valued interval of dimension n, A, are defined as

$$\begin{aligned} \mathsf{w}(A) &= \max_{i=1}^{n} \mathsf{w}(A_i) \\ \mathsf{mid}(A) &= (\mathsf{mid}(A_1), \mathsf{mid}(A_2), \ldots, \mathsf{mid}(A_n)). \end{aligned}$$

Hereafter, we will use the term interval to refer to both intervals and vector-valued intervals; the distinction will be clear from the context.

An *inclusion function* for a function f, written $\Box f$, produces an interval bound on the output of f over an interval representing its input domain. Mathematically, for all intervals X in the domain of f, if a point x is in the input interval X then $f(x)$ is contained in the output interval $\Box f(X)$; i.e.,

$$x \in X \Rightarrow f(x) \in \Box f(X) \qquad \text{for all } x \in X.$$

Much more information about inclusion functions and their properties can be found in the literature (see, for example, [MOOR79, ALEF83,RATS88]). Section 8 and the Appendices discuss ways to create inclusion functions given the functions they are to bound.

3.2 Constrained Minimization Algorithm

The constrained minimization problem involves finding the global minimizers[7] of an objective function $f: \mathbf{R}^n \rightarrow \mathbf{R}$ for all points that satisfy a constraint function $F: \mathbf{R}^n \rightarrow \{0, 1\}$.

For the case of computing collisions between parametric surfaces, we have the following variables, objective function, and constraint function:

$$\begin{aligned} x &\equiv (u_1, v_1, u_2, v_2, t) \\ f(x) &\equiv t \\ F(x) &\equiv (C(u_1, v_1, u_2, v_2, t) = 0) \text{ and} \\ &\quad\ (D(u_1, v_1, u_2, v_2, t) \geq 0) \end{aligned}$$

A region in the minimization algorithm is a 5D interval vector of the form

$$\begin{aligned} X &\equiv (U_1, V_1, U_2, V_2, T) \\ &\equiv \left(\left[u_1^l, u_1^u\right], \left[v_1^l, v_1^u\right], \left[u_2^l, u_2^u\right], \left[v_2^l, v_2^u\right], \left[t^l, t^u\right]\right) \end{aligned}$$

where the superscripts l and u denote lower and upper bounds. The relevant inclusion functions are[8]

$$\begin{aligned} \Box f(X) &\equiv \left[t^l, t^u\right] \\ \Box F(X) &\equiv \begin{cases} [0, 1], & \text{if } \mathsf{lb}\,\Box C(X) \leq 0,\ \mathsf{ub}\,\Box C(X) \geq 0, \\ & \quad \text{and } \mathsf{ub}\,\Box D(X) \geq 0 \\ [0, 0], & \text{otherwise} \end{cases} \end{aligned}$$

where $\Box C$ is an inclusion function for the collision equality constraint C, and $\Box D$ is an inclusion function for the incoming inequality constraint D.

The algorithm in Figure 6 finds solutions to the constrained minimization problem in a specified region X_0. The algorithm uses a priority queue to order regions based on the upper bound of the objective function. Regions bounding the set of global minimizers

[7]The global minimizers are the domain points at which the global minimum of the objective function is achieved, subject to the constraints.

[8]The terminology $\mathsf{lb}\,\Box C(X) \leq 0$ denotes that $\mathsf{lb}\,\Box C_i(x) \leq 0$ for *each* component interval i of $\Box C(X)$ (and similarly for upper bounds).

```
Minimize($\Box f$,$\Box F$,$A$,$X_0$,$\Box d$,$\epsilon$,$\delta$)
place initial region $X_0$ on priority queue $L$
initialize $f$'s upper bound $u \leftarrow +\infty$
initialize solution set $S \leftarrow \emptyset$
initialize singular solution set $\bar{S} \leftarrow \emptyset$
while $L$ is nonempty
    get next region $Y$ from $L$
    if lb $\Box f(Y) > u + \epsilon$ discard $Y$
    else if lb $\Box f(Y) > u - \epsilon$ and
        there exists $S_i \in \bar{S}$ such that $||\Box d(Y) - \Box d(S_i)|| < \delta$
        then discard $Y$
    else if $Y$ satisfies acceptance criteria $A$ then
        add $Y$ to solution list $S$
        if $Y$ doesn't contain a unique feasible point
            add $Y$ to $\bar{S}$
        endif
        $u \leftarrow$ min($u$, ub $\Box f(Y)$)
        delete from $S$ and $\bar{S}$ all $S_i \ni$ lb $\Box f(S_i) > u + \epsilon$
    else
        subdivide $Y$ into regions $Y_1$ and $Y_2$
        for $Y_i \in \{Y_1, Y_2\}$
            evaluate $\Box F$ on $Y_i$
            if $\Box F(Y_i) = [0, 0]$ discard $Y_i$
            evaluate $\Box f$ on $Y_i$
            if lb $\Box f(Y_i) > u + \epsilon$ discard $Y_i$
            insert $Y_i$ into $L$ according to ub $\Box f(Y_i)$
        endfor
    endif
endwhile
```

Figure 6: Global Constrained Minimization Algorithm: This algorithm finds the global minimizers of an objective function f, with constraints F, acceptance criteria A, initial region X_0, solution distance mapping function d, simultaneity threshold ϵ, and solution separation distance δ.

are subdivided until they are rejected or satisfy the acceptance criteria, A, and are accepted as solutions. It halts with an empty list of solutions if there are no solutions to the constraint function in X_0, or a list of regions, S, representing the set of global minimizers of the constrained minimization problem.

The variable u is a progressively refined least upper bound for the global minimum of the objective function. If we were only looking for a single collision point, we could halt the algorithm immediately after finding the first solution. To find collisions at multiple points of contact, the algorithm must be continued until the priority queue is empty. The variable u helps to prune the search after finding the first solutions.

Selecting Finite Sets of Points from Contact Manifolds The parameters ϵ, δ, and $\Box d$ allow the algorithm to select a finite set of regions distributed "uniformly" within the set of global minimizers, when this set is not finite. The parameter ϵ is the *simultaneity threshold*, which specifies how close the value of the objective function must be for two points to be considered global minimizers. For collisions, ϵ specifies how close in time two events must be in order to be considered simultaneous. The parameter δ is the *solution separation distance*, which specifies how far apart two accepted regions must be to be accepted as separate solutions. The parameter $\Box d$ is an inclusion function for the mapping which takes points in parameter space to points in whatever space we desire distances to be compared. We call the function d the *solution distance mapping function*.

As the algorithm progresses, it maintains two solution lists, S and $\bar{S}$. S contains all accepted regions. We call $\bar{S}$ the *singular solution set*. The elements of S not in $\bar{S}$ are regions in which the existence of a unique feasible point has been verified. The statements

```
if lb $\Box f(Y) > u - \epsilon$ and there exists $S_i \in \bar{S}$
    such that $||\Box d(Y) - \Box d(S_i)|| < \delta$
then discard $Y$
```

check that the region Y is not too close to regions already accumulated onto $\bar{S}$. Note that the test lb $\Box f(Y) > u - \epsilon$ is critical to ensure that Y doesn't have an objective function value small enough to invalidate all the currently accepted regions.[9]

We use two lists, S and $\bar{S}$, so that in the case that the global minimizers form a finite set of points, the algorithm can find all such points without discarding some based on distance to those already found. The algorithm is therefore able to resolve multiple isolated collisions that happen in a small area, regardless of the value of δ.[10]

Ordering Based on Upper Bounds Constrained minimization algorithms that have appeared before [RATS88,SNYD92c] order regions based on the *lower bound* of the objective function. We use the upper bound to make tractable computing solutions on a contact manifold.

At any time, the union of all regions on the priority queue forms a bound on the set of global minimizers of the constrained minimization problem. As the algorithm progresses, regions are subdivided or rejected, so that the regions which remain on the priority queue become a tighter bound on this set. Because of the inclusion monotonicity property of inclusion functions,[11] as regions on the queue shrink, the computed lower bound on the objective function tends to increase and the upper bound tends to decrease.

Assume the set of global minimizers forms a continuous manifold rather than a finite collection of isolated points, as shown Figure 1. If the priority queue is ordered using lower bounds, when a given region is subdivided, its children will generally have larger lower bounds for the objective function, and will be placed in the priority queue behind less highly subdivided regions. A breadth first traversal tends to result, with less highly subdivided regions examined first. If we have a whole manifold of global minimizers and stringent acceptance criteria, we will have to compute a huge number of tiny regions bounding the entire solution manifold before even the first region is accepted as a solution.

By ordering based on the upper bound, more highly subdivided regions tend to be examined first because they tend to have smaller upper bounds. We quickly get to a region which is small enough to satisfy the acceptance criteria. This allows our upper bound u to be updated. It also allows regions to be accumulated onto our singular list $\bar{S}$. Regions that are too close to any member of $\bar{S}$ can then be eliminated, making it possible to find a distribution of points on the contact manifold without undue computation.

Acceptance Criteria The constraint inclusion function, $\Box F(X)$, because it contains an equality constraint, returns either [0, 0] (i.e.,

[9] If a region can possibly have a feasible point with a value of f less than ϵ from the value of f in regions on $\bar{S}$, we should not reject it just because it is close with respect to the function d to these regions. The algorithm might then discard a global minimizer because of its closeness to regions which are possibly far from the global minimizer, in terms of bounds on the objective function.

[10] One problem with this technique is that if the collisions happened at a contact manifold *and* a finite number of additional isolated points, the algorithm may discard some of the isolated points because of the closeness criterion. We consider this problem minor since the set of global minimizers is infinite and the algorithm must chose a subset anyway.

[11] An inclusion function, $\Box f$, is inclusion monotonic if $Y \subset X \Rightarrow \Box f(Y) \subset \Box f(X)$. In practice, the standard ways of constructing inclusion functions generate inclusion monotonic inclusion functions.

the constraint is satisfied nowhere in X) or [0, 1] (i.e., the constraint *may* be satisfied in X). We must resort to other means to determine if the constraint is *actually* satisfied. Section 5 discusses conditions which guarantee that a region contains a unique solution to the equality constraints. These conditions can therefore be used as acceptance criteria in the algorithm, which we call the *isolated point acceptance criteria.* They also allow the upper bound u to be updated via

$$u \leftarrow \min(u, \text{ub}\,\Box f(Y))$$

since Y is guaranteed to contain a feasible point.

The algorithm also makes use of *emergency acceptance criteria* which do not guarantee a unique solution but are guaranteed to be satisfied for regions of small enough size.[12] The simplest such criterion is $\text{w}(X) < \epsilon$; a better one is $\text{w}(\Box S(X)) < \epsilon$ where $\Box S$ is the parametric mapping of one of the colliding surfaces. Regions which are accepted via the emergency acceptance criteria are inserted both onto the list of solutions, S, and the singular solutions, $\bar{S}$.

Subdivision The simplest method of subdividing candidate intervals in the minimization algorithm is bisection, in which two intervals are created by subdividing one of the input dimensions at its midpoint. Many methods can be used to select which dimension to subdivide. For example, we can simply pick the dimension of greatest width. A better alternative is to scale the parametric width by some measure of its importance to the problem we are solving. For each variable in the collision problem x_i, and a given candidate region X, we have used a scaling value s_i defined by

$$s_i \equiv \sum_{j=1}^{m} \max(|\,\text{lb}\,\Box\frac{\partial f_j}{\partial x_i}(X)|, |\,\text{ub}\,\Box\frac{\partial f_j}{\partial x_i}(X)|).$$

Here, f refers to the equality constraints (contact and tangency) of the collision problem. We then pick a dimension to subdivide, i, such that the scaled width $s_i\,\text{w}(X_i)$ is largest.

Given a candidate interval, techniques also exist which allow us to compute a smaller interval which can possibly contain feasible points of the constraint. These methods and how they can be added to our simple minimization algorithm are discussed in Section 4. Even more sophisticated subdivision methods exist, such as Hansen's method which involves accumulating gaps inside candidate intervals by using infinite interval division (see [RATS88] for a full description).

Multiple Element Constrained Minimization The algorithm can easily be modified to accept an array of sets of minimization parameters $(\Box f, \Box F, A, X_0)_i$. This allows simultaneous solution of sets of problems from different pairs of surfaces, or different tangency situations for the same pair of piecewise parametric surfaces. As a result, computation is not wasted on collisions which happen after the first collision, $t > t^*$. We call this modified constrained minimization algorithm the *multiple element constrained minimization algorithm.*

Sets of minimization subproblems may be implemented by associating the array index of the appropriate minimization subproblem with each region inserted onto the priority queue, and using the appropriate indexed inclusion functions and acceptance criteria when processing the region.

[12] Although we cannot guarantee a region X contains a solution, we can guarantee that it is arbitrarily close, in the sense that $\text{w}(\Box C(X)) < \epsilon$ where $\Box C$ is an inclusion function for the collision equality constraint function C.

Avoiding Detection of Tracked Points We can add additional inequality constraints to the constraint function F in order to avoid detecting collisions which occur at contact points already being tracked. If p is such a tracked point on a surface $S(u, v, t)$, the ODE solver computes a trajectory for $p = S(\tilde{u}(t), \tilde{v}(t), t)$. We then discard all global minimizers to the constrained minimization problem which satisfy

$$||S(u, v, t) - S(\tilde{u}(t), \tilde{v}(t), t)|| < \lambda, \tag{7}$$

where λ is a constant chosen by the user. The functions $\tilde{u}$ and $\tilde{v}$ have known representations, as computed by the solver. A natural interval extension of this constraint involving an inclusion function for S is then included in the constraint inclusion $\Box F$. An additional constraint is added for each tracked point.

4 Interval Newton Methods

In order to more quickly refine our intervals towards the solutions of the collision equality constraint $C = 0$, we make use of an interval Newton method. Interval Newton methods are applicable to the general problem of finding zeroes of a differentiable function $f\colon \mathbf{R}^n \rightarrow \mathbf{R}^m$ in an interval $X \subset \mathbf{R}^n$. They allow us to find an interval bound on the set

$$X^* = \{x \in X \mid f(x) = 0\}$$

Let $Z(X)$ be such a bound (i.e., $X^* \subset Z(X)$). We can reduce the size of our candidate region X by[13]

$$X' = X \bigcap Z(X)$$

In particular, $Z(X) \bigcap X = \emptyset$ implies that X contains no solutions. We call the operator $Z(X)$ the *interval Newton operator.*

Since X can only decrease in size after it is intersected with $Z(X)$, this procedure can be applied iteratively to produce smaller and smaller regions, as in

$$X_{i+1} = (X_i \bigcap Z(X_i))$$

Note however that a smaller region is not *necessarily* produced. Interval Newton methods should therefore be combined with bisection. When interval Newton iteration is effective at reducing the size of X its use is continued. Otherwise, bisection subdivision is performed.

The following sections present three methods for computing $Z(X)$:

- use of the Krawczyk-Moore form (Section 4.1)
- use of the interval inverse (Section 4.2.1)
- use of matrix iteration (Section 4.2.2)

In each case, we modify the constrained minimization algorithm from Figure 6 by replacing the subdivide step with the code shown in Figure 7.

4.1 Fixed Point Methods: the Krawczyk–Moore Form

The familiar (point-wise) Newton's method is used to converge on the solution to a system of equations $f(x) = 0$ where $f\colon \mathbf{R}^n \rightarrow \mathbf{R}^n$.

[13] $A \bigcap B$ denotes the interval formed by the intersection of the intervals A and B.

```
Newton(Y)
compute interval Newton step on Y
if step succeeds then
    $Y' \leftarrow Y \bigcap Z(Y)$
    if $Y' = \emptyset$, discard $Y$
        (proceed with next region)
    else if $Y'$ is sufficiently smaller than $Y$, insert $Y'$ into $L$
        (proceed with next region)
    else subdivide $Y'$
        (continue with $Y$ replaced by $Y'$)
else subdivide $Y$
        (continue with $Y$)
```

Figure 7: Interval Newton Modification to the Constrained Minimization Algorithm: The above algorithm replaces the **subdivide** step in the algorithm of Figure 6.

The method starts with an initial "guess" at a solution x_0 and iterates via

$$x_{i+1} = p(x_i)$$

where

$$p(x) = x - Yf(x).$$

Y is a nonsingular $n \times n$ matrix which in straightforward Newton's method is the inverse of the Jacobian matrix of f at x, i.e.

$$Y \equiv J^{-1}(x)$$

Under certain conditions, this iterative procedure converges to a fixed point x^*. If convergence is achieved, then the fixed point x^* is a solution since

$$p(x^*) = x^* \Longleftrightarrow f(x^*) = 0$$

because Y is nonsingular.

An interval analog of this method may be developed. Let X be an interval in $\mathbf{R}^n$ in which zeroes of f are sought. We require a bound on X^*. But

$$\begin{aligned} X^* &\equiv \{x \in X \mid f(x) = 0\} \\ &= \{x \in X \mid p(x) = x\} \subset (\Box p(X) \bigcap X) \end{aligned}$$

where $\Box p(X)$ is an inclusion function for the Newton operator $p(x)$.

The Krawczyk–Moore form, $K(X, c, Y)$, provides the necessary inclusion function for the Newton operator $p(x)$. It is simply a mean value form for p (see [SNYD92c] for a discussion of the mean value form) given by

$$K(X, c, Y) \equiv c - Yf(c) + (I - Y\Box J(X))(X - c)$$

where I is the $n \times n$ identity matrix, $\Box J(X)$ is an inclusion function for the Jacobian of f evaluated on X, and c is any point in X. Note that the vector addition and subtraction and the matrix/vector multiplication operations used in K must be computed using interval arithmetic.

We therefore have

$$X^* \subset (X \bigcap K(X, c, Y))$$

for any $c \in X$ and nonsingular matrix Y. Thus, $K(X, c, Y)$ can be used as an interval Newton operator. Fairly good results can be achieved with $c = \text{mid}(X)$ and $Y = J^{-1}(c)$ [TOTH85, MITC92].

In our research, we have found a different method to be superior, described in the next section.

4.2 Linear Interval Equation Methods

A second method for finding an interval bound on X^* involves solving the interval analog of a linear equation.

Let the coordinates of x be $x_1, x_2, \ldots, x_n$. By the Mean Value Theorem, given a $c \in X$, for each $x \in X$, there exist n points, $\xi_1, \xi_2, \ldots, \xi_n$ such that

$$f(x) = f(c) + J(\xi_1, \ldots, \xi_n)(x - c),$$

where the jacobian matrix J is given by

$$J_{ij}(\xi_1, \xi_2, \ldots, \xi_n) = \frac{\partial f_i}{\partial x_j}(\xi_i)$$

and where each $\xi_i \in X$. Let $\Box J$ be an inclusion function for the Jacobian matrix of f, i.e.,

$$\Box J(X) \equiv \left\{ J \mid J_{ij} \in \Box\frac{\partial f_i}{\partial x_j}(X) \right\}$$

If x is a zero of f, then there exists $J \in \Box J(X)$ such that

$$f(x) = 0 = f(c) + J(x - c).$$

Therefore, if $Q(X)$ is the set of solutions

$$Q(X) \equiv \{x \mid f(c) + J(x - c) = 0 \text{ for some } J \in \Box J(X)\},$$

then $Q(X)$ contains all zeroes of f in X.

To compute an interval bound, Z, on $Q(X)$, let $y = x - c$, and let Z' be an interval bound on the set

$$\{y \mid Jy = -f(c) \text{ for some } J \in \Box J(X)\}.$$

Then the interval Z defined using interval addition as

$$Z \equiv Z' + [c, c],$$

is an interval bound on $Q(X)$. Thus, computing the interval Newton bound Z can be accomplished by solving an interval linear equation of the form

$$Mx = b$$

where $M \equiv \Box J(X)$ is an $n \times n$ interval matrix, and $b \equiv -f(c)$ is an interval vector.[14] Stated another way, we require a bound on the set

$$Q(M, b) \equiv \{x \mid \exists \mathcal{M} \in M, \beta \in b \text{ such that } \mathcal{M}x = \beta\}.$$

The next two sections discuss two methods for solving these interval linear equations.

4.2.1 Solving the Interval Linear Equation with the Interval Inverse

One method to bound the set of solutions to the interval linear equation involves computing the interval inverse. We seek a bound on $Q(M, b)$: the set of solutions, x, for $Mx = b$. If M is an $n \times n$ interval matrix, an interval that bounds $Q(M, b)$ is

$$Z \equiv M^{-1}b$$

[14] In this case, the interval b has a lower bound equal to its upper bounds in each coordinate (called a *point* interval), neglecting inaccuracies in the computation of f.

where M^{-1} is the *interval inverse* of the interval matrix. Assuming M contains no singular matrices, the interval inverse is an interval bound on the set

$$\{m^{-1} \mid m \in M\}$$

A simple way of computing the interval inverse is to use the interval analog of LU decomposition. That is, we take the LU decomposition algorithm [PRES86, pages 31–38] and replace all arithmetic operations with their corresponding interval arithmetic counterparts (see [MOOR79] for a discussion of interval arithmetic). If at any point in the iteration we attempt to divide by an interval which contains zero, then we cannot compute the interval inverse, and the interval Newton step fails (but see the next section for a way to reduce the size of candidate regions without using the interval inverse). After enough iterations of the constrained minimization algorithm, and assuming the conditions discussed in Section 5 hold, candidate regions are usually small enough to make this technique effective.

4.2.2 Solving the Interval Linear Equation with Matrix Iteration

Another method to bound the set of solutions to the interval linear equation involves matrix iteration. The algorithm we present here requires an initial bound on the set of solutions; that is it finds a bound on the set $Q(M,b) \bigcap X$ where X is a given interval. The algorithm is therefore effective at reducing the size of a candidate interval in which solutions to an equality constraint are sought, but cannot be used to verify solution existence using the theorems in Section 5.[15]

Figure 8 contains the code for Linear_Solve, which finds bounds on the solution to the linear interval equation. Linear_Solve is based on the observation that the i-th equation of the linear system $Mx = b$:

$$M_{i1}x_1 + M_{i2}x_2 + \cdots + M_{in}x_n = b_i$$

implies, for each j such that $M_{ij} \neq 0$, that

$$x_j = \frac{b_i - \sum_{k \neq j} M_{ik}x_k}{M_{ij}}.$$

The interval analog of this equation may therefore be used for each interval matrix entry, M_{ij}, which does not contain 0 to find a bound on one of the variables x_j. This bound is intersected with the old bound on x_j yielding an interval which is possibly smaller but no larger than it was. Reducing the size of one interval may then further reduce the sizes of others as the iteration proceeds. Note that the algorithm does not halt when an interval element of M contains 0; it just proceeds to the next element which excludes 0.

An important property of Linear_Solve is that it can be applied to a nonsquare linear equation,[16] and is therefore useful in the "overconstrained" equality constraint for implicit surfaces, and the vertex-to-edge and vertex-to-vertex tangency situations of piecewise parametric surfaces (see Appendix A). Linear_Solve can be applied in many situations where LU decomposition fails because of the singularity of the interval Jacobian matrix. Even when the Jacobian matrix is singular *at the solution point*, Linear_Solve is usually effective at reducing the widths of some of the input variables. These features are critical in making the singular situations described in Section 5 computationally tractable.[17]

```
Linear_Solve($M,b,x$)
repeat
    loop through rows of $M$ ($i = 1,2,\ldots,m$)
        loop through columns of $M$ ($j = 1,2,\ldots,n$)
            if $0 \notin M_{ij}$ then
                $x'_j \leftarrow (b_i - \sum_{k \neq j} M_{ik}x_k)/M_{ij}$
                $x_j \leftarrow x'_j \bigcap x_j$
                if $x_j = \emptyset$ return no solution
            endif
        endloop
    endloop
while there is sufficient improvement in $x$
```

Figure 8: Interval Linear Equation Solution Algorithm: This algorithm computes the interval Newton step (first statement of the algorithm in Figure 6).

The "sufficient improvement" condition mentioned in the algorithm can be implemented as

$$\mathrm{w}(x^{i+1}) \leq \alpha\, \mathrm{w}(x^i)$$

where a typical value of the improvement factor, α, is 0.9. Here x^i denotes the interval bound computed after i iterations of the repeat loop. Specifying a maximum number of repeat iterations also limits the amount of computation.

5 Termination Criteria

Two theorems in interval analysis specify conditions under which a square system of equations contains a unique solution in a region.[18]

Theorem 1 (Krawczyk–Moore Existence) If $K(X,c,Y) \subset X$, $K(X,c,Y) \neq \emptyset$, and $\|I - Y\Box J(X)\| < 1$, then there is a unique root in X, and pointwise Newton's method will converge to it.

Theorem 2 (Linear Interval Equation Existence) If $Q(X) \subset X$ and $Q(X) \neq \emptyset$, then there is a unique root in X.

The conditions implied by these theorems thus lead to acceptance criteria, A, for the constrained minimization algorithm. Implementation of Theorem 1's conditions is clear from the discussion in Section 4.1. To verify the conditions of Theorem 2, we use the interval inverse method discussed in Section 4.2.1. We have been able to verify solution uniqueness much earlier (i.e., in larger regions) using Theorem 2's test.

We note the conditions for Theorems 1 and 2 can only be verified when the determinant of the Jacobian of the equality constraint function, C, is nonzero in some neighborhood of the solution. For collision detection, the Jacobian determinant is zero at a solution to the contact and tangency constraints in the following situations:

- the contacting surfaces become tangent but never interpenetrate. They can even stay tangent for an interval of time.

[15]This is because, unlike the technique presented in Sections 4.2.1, this technique does not bound $Q(M,b)$ directly, but instead bounds $Q(M,b) \bigcap X$.

[16]That is, the number of equations, m, is unequal to the number of variables, n.

[17]We prefer the method of matrix iteration described here to a faster method (the interval analog of Gauss-Seidel iteration) which involves solving only for the diagonal matrix elements after a preconditioning step (see [RATS88]). This method requires a square system of equations, and will fail when the Jacobian matrix is singular at the solution. Interval computations in the preconditioning step also have the effect of increasing the size of the solution set $Q(M,b)$ even before any iteration takes place.

[18]See [TOTH85] for a proof sketch and references for Theorem 1, [SNYD92b] for a proof of Theorem 2.

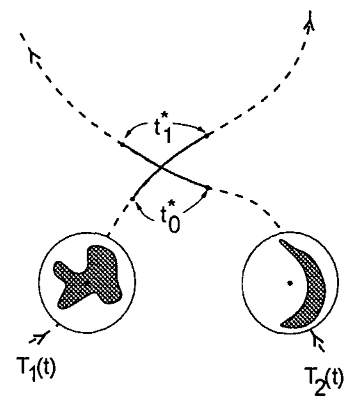


Figure 9: Bounding Sphere Collision Time Bound

- the surfaces contact instantaneously on a curve or higher-dimensional region.
- both — the surfaces contact at an infinite set of points through an interval of time.

In these cases, we can not verify that a unique solution exists and must resort to heuristic criteria (the "emergency" criteria of the constrained minimization algorithm). That is, we consider surfaces to have collided when a bound on C is sufficiently small in a region.[19]

If a solution is on the boundary of a candidate region, we note that the conditions of either theorem will be difficult to verify since the set result (e.g., $Q(X)$) will always slightly spill out of the original region X. To solve this problem, we should slightly increase the region examined for acceptance, X, via

$$X' \equiv c + (X - c)(1 + \gamma)$$

where γ is a small constant (like .1). We then can perform the test on this bigger region X'. Then, even if the solution is on the boundary of X, the theorem conditions will eventually be satisfied if the Jacobian of C is nonsingular in a neighborhood of the solution.

6 A Simple Bound for the Time of Collision

We can save time in the collision algorithm by using a fast algorithm to reduce the time interval over which collisions are searched. This time bound may also tell us, with a minimum of computation, if the two surfaces fail to intersect, obviating the need for further computation. The test presented here uses 1D minimization (only over time) rather than minimization over the 4 or 5 dimensional space required to solve the full problem.

As shown in Figure 9, we compute two bounding spheres around each of our parametric surfaces. In the case of rigid motion, this may be computed beforehand as a preprocessing step. We specify two points for each parametric surface O_1 and O_2 about which to compute a bounding sphere. These points should be chosen to minimize the size of the resulting bound; using the center of mass of the surface is a good choice. The bounding radii are:

$$R_1 \equiv \max_{(u_1,v_1)} \|s_1(u_1,v_1) - O_1\|$$
$$R_2 \equiv \max_{(u_2,v_2)} \|s_2(u_2,v_2) - O_2\|.$$

[19]This implies, because of the contact constraint, that the surfaces come within a specified constant.

Note that these bounding radii can be computed using a 2D unconstrained minimization problem, for which the algorithm of Section 3.2 is suitable.

Using the constrained minimization algorithm, this time on a simple 1D problem, we then find a bound on the time of collision via

$$t_0^* \equiv \underset{t\in[t_0,t_1]}{\text{minimum}}\{\ t \mid \|T_1(t) + O_1 - T_2(t) - O_2\| \le R_1 + R_2\}$$
$$t_1^* \equiv -\underset{t\in[t_0,t_1]}{\text{minimum}}\{-t \mid \|T_1(t) + O_1 - T_2(t) - O_2\| \le R_1 + R_2\}.$$

We can then replace the $[t_0, t_1]$ interval in the full collision minimization problem for that pair of surfaces with $[t_0^*, t_1^*]$, or cull the pair of surfaces if no solutions to the 1D problem are found.

7 The Full Collision Algorithm

The complete algorithm for detecting collisions can now be described. The following discussion pertains to a set of parametric surfaces; a similar algorithm can be developed for the case of implicit surfaces or parametric/implicit combinations. We are given a set of solids defined by a parametric boundary representation, and a time interval in which to detect collisions, $[t_0, t_1]$. The following steps summarize the final algorithm:

1. Detect pairs of objects which can possibly collide. For this step, we bound each time-varying surface by evaluating an inclusion function for its time-varying mapping over $[t_0, t_1]$. More precisely, a bounding box through time on the surface $S(u, v, t)$ is given by

$$\Box S([0,1],[0,1],[t_0,t_1])$$

assuming S is evaluated on the unit square.[20] We can then test whether any of the resulting bounding boxes intersect using highly efficient algorithms from computational geometry [SIX82]. All pairs of bounding boxes which do intersect must be processed further; the rest are culled.

2. For rigid bodies, additional object pairs can be culled using the bounding sphere test of Section 6. A variant of this test can also be used for deformable surfaces.

3. If any pairs of objects remain to be processed, we must invoke the full constrained minimization algorithm. Here, we distinguish between "free" objects and objects already in continuous contact, whose contact points are being tracked with the ODE solver. For objects already in continuous contact, additional constraints are added (Section 3.2) to prevent re-detection of the tracked points. All such problems are placed on the initial priority queue of the multiple element constrained minimization algorithm.

4. We use local methods, such as Newton's method, to converge to the actual collision point in each solution region which contains an isolated collision (i.e., for which the interval existence and uniqueness test of Section 5 succeeded). We arbitrarily choose the midpoint as the collision point for the rest of the solution regions (termed singular solutions in Section 3).

[20]Note that for rigid surfaces we can cache a bounding box on the time-independent rigid surface and compute only a bound over time on the resulting rotated and translated bounding box.

8 Implementing Inclusion Functions

The collision detection algorithm depends on inclusion functions for the time-varying surfaces and their various derivatives. Note that the equality constraint for the parametric surface case (Equation 2) involves derivatives of the time-varying surface mappings $S_i(u_i, v_i, t)$. The interval Newton method then requires an additional derivative of the equality constraint with respect to each of the independent variables. Interval analysis provides the necessary theory for constructing inclusion functions for these functions.

For simple polynomial surfaces (e.g., bicubic patches or algebraic surfaces) interval arithmetic suffices to provide an inclusion function for the time-independent surface. Toth [TOTH85] has presented efficient inclusion functions for Bezier surfaces. Mitchell and Hanrahan have proposed a simple stack-based representation of surfaces which allows generation of inclusion functions for the surface and its derivatives [MITC92]. Inclusion functions for more complicated surfaces and their derivatives can also be constructed. We have used the system described in [SNYD92a,SNYD92b], which automates the construction of inclusion functions (and inclusion functions for the derivatives) of any functions formed by the composition of a quite powerful set of symbolic operators.

For physical simulations, the ODE solver computes a representation of the time behavior of the surfaces. The solver may directly compute a continuous representation or it may be later reconstructed by point sampling the solver's results, typically producing a polynomial. Appendix B discusses a method to bound Chebyshev polynomials.

9 Results

We have successfully tested this method on a series of collision detection examples, including both rigidly moving and deforming objects. For example, Figure 12 shows the results of a difficult collision detection run in which the contact manifold forms a series of disjoint 2D regions. A collection of 59 points was generated in the contact region with a simultaneity threshold of 0.001 and solution separation distance of 0.04, using 28704 iterations and 88.81 CPU seconds.[21] While the running time may seem large, the problem itself is sufficiently difficult that its running time exceeded our threshold of 8 CPU hours without the use of *every* new technique presented in this paper: adding the tangency constraint (rather than using the contact constraint alone), sorting by upper bound in the constrained minimization algorithm (rather than by lower bound), and using Linear_Solve for the interval Newton step (rather than the Krawczyk–Moore operator). Figure 1, 5, and 12–16 show the results of the algorithm for several different time-varying shapes.

The table in Figure 10 compares running times for a second example involving two rotating and translating bumpy parametric surfaces which collide at an isolated point. Several solution methods are compared: LEQN (interval Newton using the linear equation solution techniques of Section 4.2), KM (interval Newton using the Krawczyk–Moore operator), NIN (without interval Newton), and NTAN (without the tangency condition). Since the collision occurs at an isolated point, both the LEQ and KM methods were able to accept a single solution region by verifying the solution existence and

Running Times		
Example	Iterations	CPU (secs)
LEQN	6331	32.67
KM	10087	148.28
NIN,γ=1e-3	17395	8.58
NIN,γ=1e-4	29921	15.46
NIN,γ=1e-5	40127	21.52
NIN,γ=1e-6	48187	23.25
NIN,NTAN,γ=1e-3	52307	14.59
NIN,NTAN,γ=1e-4	587711	169.87
NIN,NTAN,γ=1e-5	3822605	1207.46

Figure 10: Table of Results for Various Methods: see Section 9

uniqueness test. The other methods required an accuracy parameter for acceptance; we used the simple criterion $w(X) < \gamma$.

Because we used a prototype system to gather the data, we emphasize the importance of iteration count data over CPU time. Our system requires the traversal of a complicated data structure for each inclusion function evaluation which overwhelms the floating point computation actually needed in the function. The interval Newton methods are sensitive to this bias, since their implementation required many symbolic operators. We believe the iteration counts shown here to be a reasonable measure of expected running time, if the inclusion functions are hand-coded for the surfaces of interest.

10 Conclusions

We have presented a robust interval algorithm that can detect collisions between complex curved surfaces. The algorithm handles a greater range of situations than previous algorithms. It detects both isolated collision points and collision points on contact manifolds. It can avoid detection of points close to a set of tracked points with specified trajectories. It efficiently handles detection of simultaneous collisions between sets of moving objects. The technique is practical for simulations involving large numbers of moving and deforming objects (see Figures 15 and 16).

We draw several conclusions from our experimental results. First, interval methods, such as [VONH90] and [DUFF92], which do not make use of the interval Newton method or the tangency condition soon become impractical as we increase the accuracy parameter (refer to the NIN,NTAN lines of the table in Figure 10). Interval Newton iteration combined with the tangency condition (especially using the interval linear equation approach) is very effective at reducing computation. Second, our method can solve the difficult problem of detecting collision points on a contact manifold. We have found the methods described here to be indispensable, including the idea of the tangency constraint, the constrained minimization algorithm discussed in Section 3, and the interval linear equation approach to interval Newton iteration.

We note that many areas for improvement remain. Sorting by lower bound of the objective function rather than by upper bound is more efficient for isolated point collisions. We have noted an efficiency gain of a factor of from 1 to 10 in using the lower bound for such cases. On the other hand, sorting by lower bound is completely impractical for detecting collisions on a contact manifold. If we know the nature of the collision solution set a priori, we can choose the appropriate method. Alternatively, combining the two approaches, perhaps by "racing" them in parallel on the same problem, may decrease the average running time. We are studying several ways to increase efficiency that involve more optimally

[21] The term *iteration* refers to an evaluation of the inclusion functions $\Box f$ and $\Box F$ (objective function and constraint function) in the constrained minimization algorithm. All CPU times are measured on a HP 9000 Series 750 computer.

choosing the next dimension to subdivide, and determining a subdivision location other than the midpoint.

Acknowledgments

We wish to thank Mark Montague and Allen Corcorran for their production help. Thanks also go to the Siggraph reviewers for their careful reading, suggestions, and bug fixes. This work was supported in part by grants from Apple, DEC, Hewlett Packard, and IBM. Additional support was provided by NSF (ASC-89-20219), as part of the NSF/DARPA STC for Computer Graphics and Scientific Visualization. All opinions, findings, conclusions, or recommendations expressed in this document are those of the author and do not necessarily reflect the views of the sponsoring agencies.

References

[ALEF83] Alefeld, G., and J. Herzberger, *Introduction to Interval Computations,* Academic Press, New York, 1983.

[BARA89] Baraff, David, "Analytical Methods for Dynamic Simulation of Non-penetrating Rigid Bodies," Computer Graphics, 23(3), pp. 223-232, July 1989.

[BARA90] Baraff, David, "Curved Surfaces and Coherence for Non-penetrating Rigid Body Simulation," Computer Graphics, 24(4), pp. 19-28, August 1990.

[BARA91] Baraff, David, "Coping with Friction for Non-penetrating Rigid Body Simulation," Computer Graphics, 25(4), pp. 31-39, July 1991.

[BARA92] Baraff, David, and A. Witkin, "Dynamic Simulation of Non-penetrating Flexible Bodies," Computer Graphics, 26(2), pp. 303-308, July 1992.

[DUFF92] Duff, Tom, "Interval Arithmetic and Recursive Subdivision for Implicit Functions and Constructive Solid Geometry," Computer Graphics, 26(2), July 1992, pp. 131-138.

[META92] Metaxas, Dimitri, and D. Terzopoulos, "Dynamic Deformation of Solid Primitives with Constraints," Computer Graphics, 26(2), pp. 309-312, July 1992.

[MITC92] Mitchell, Don, and P. Hanrahan, "Illumination from Curved Reflectors," Computer Graphics, 26(2), July 1992, pp. 283-291.

[MOOR66] Moore, R.E., *Interval Analysis,* Prentice Hall, Englewood Cliffs, New Jersey, 1966.

[MOOR79] Moore, R.E., *Methods and Applications of Interval Analysis,* SIAM, Philadelphia.

[MOOR80] Moore, R.E., "New Results on Nonlinear Systems," in *Interval Mathematics 1980,* Karl Nickel, ed., Academic Press, New York, 1980, pp. 165-180.

[MOOR88] Moore, M. and Wilhelms, J., "Collision Detection and Response for Computer Animation," Computer Graphics, 22(4), pp. 289-298, August 1988.

[PRES86] Press, W. H., B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes,* Cambridge University Press, Cambridge, England, 1986.

[RATS88] Ratschek, H. and J. Rokne, *New Computer Methods for Global Optimization,* Ellis Horwood Limited, Chichester, England, 1988.

[SCLA91] Sclaroff, Stan, and A. Pentland, "Generalized Implicit Functions for Computer Graphics," Computer Graphics, 25(4), pp. 247-250, July 1991.

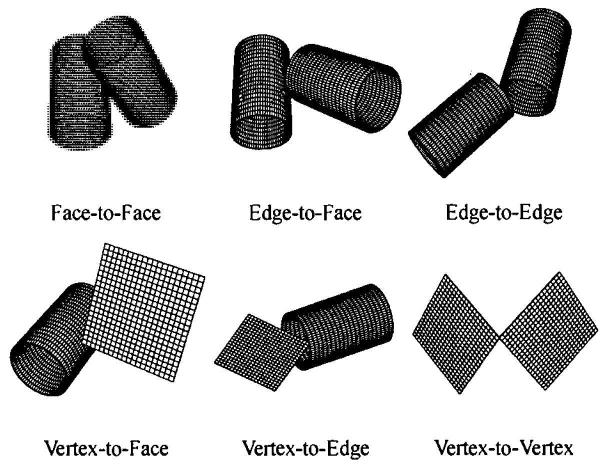


Figure 11: Types of Collision Tangency

[SIX82] Six, H.W., and D. Wood, "Counting and Reporting Intersections of *d*-Ranges," IEEE Transactions on Computers, C-31(3), March 1982, pp. 181-187.

[SNYD92a] Snyder, John, and J. Kajiya, "Generative Modeling: A Symbolic System for Geometric Modeling," Computer Graphics, 26(2), pp. 369-378, July 1992.

[SNYD92b] Snyder, John, *Generative Modeling for Computer Graphics and CAD: Symbolic Shape Design Using Interval Analysis,* Academic Press, Cambridge, MA, July 1992.

[SNYD92c] Snyder, John, "Interval Analysis for Computer Graphics," Computer Graphics, 26(2), pp. 121-130, July 1992.

[TOTH85] Toth, Daniel L., "On Ray Tracing Parametric Surfaces," Computer Graphics, 19(3), July 1985, pp. 171-179.

[VONH90] Von Herzen, B., A.H. Barr, and H.R. Zatz, "Geometric Collisions for Time-Dependent Parametric Surfaces," Computer Graphics, 24(4), August 1990, pp. 39-48.

A Collision Constraints for Piecewise Parametric Surfaces

A piecewise surface is composed of a set of smooth *faces*, a set of *edges* where these faces meet, and a set of *vertices* or points where edges meet. Edges form the 1D boundaries over which the surface is not smooth; vertices are the 0D boundaries between smooth edge curves. A data structure containing the faces, edges, and vertices of a solid is called its *boundary representation.* For example, the boundary representation of a cylindrical solid contains three faces: one cylinder and two circular endcaps, two edges where the cylinder and endcap meet, but no vertices.

To detect collisions between two piecewise surfaces, we must search for collisions between each pair of faces, between edges and faces, between vertices and faces, etc. The constraints governing collisions are different in each of these cases, which we call *tangency situations.* There are 6 types of tangency conditions in a collision between piecewise surfaces as shown in Figure 11. Constraints for the face-to-face tangency situation are identical to the constraints discussed in Section 2.1. The following paragraphs discuss the other tangency situations.

We must combine all the constrained minimization problems for the various possible types of tangency situations. For example, if the surfaces are a pair of cylindrical solids, we obtain 25 separate constrained minimization subproblems: 9 face-to-face problem, 12 edge-to-face problems, and 4 edge-to-edge problems. Two tori require only a single face-to-face problem. Each problem is then solved simultaneously using the multiple element constrained minimization algorithm.

Edge-to-Face For the *edge-to-face* case, we have an edge curve, $C(s,t)$, which forms a boundary of a surface, $S_a(u_a, v_a, t)$, and another surface, $S(u,v,t)$. The edge curve is typically formed by evaluating a parametric surface S_a at a specific value for either the u or v parameter, e.g.

$$C(s,t) \equiv S_a(s, v^{\text{fixed}}, t)$$

where v^{fixed} is a constant set at one of the extremes of the v interval over which S_a is evaluated. The contact constraint for edge-to-face collisions is

$$C(s,t) - S(u,v,t) = 0 \tag{8}$$

and the tangency constraint by

$$\frac{\partial C}{\partial s}(s,t) \cdot N(u,v,t) = 0 \tag{9}$$

where N is the time-varying normal to the surface S. The edge-to-face equality constraint can be represented a system of 4 equations in 4 variables.

To define the incoming collision condition, we need to define what "outwardness" means on an edge curve. Assuming all surfaces form the valid boundaries of a closed solid, the edge curve $C(s,t)$ is shared between two surfaces S_a and S_b. We can therefore define two "outward" directions, given by the outward pointing normals to the shared surfaces S_a and S_b. For example, these outward directions may be defined as

$$\begin{aligned} C^{\text{outward-1}}(s,t) &\equiv N_a(s, v^{\text{fixed}}, t) \\ C^{\text{outward-2}}(s,t) &\equiv N_b(u^{\text{fixed}}, s, t) \end{aligned}$$

where N_a and N_b are the outward normal vectors of the respective surfaces.

The incoming constraint forces the relative velocity between the surface and the edge curve to be in the same direction (using a dot product test) as the surface's normal. The surface's normal must also face away from at least one of the edge curve's outward directions. The incoming constraint is:

$$\begin{aligned} &(\frac{\partial S}{\partial t}(u,v,t) - \frac{\partial C}{\partial t}(s,t)) \cdot N(u,v,t) \geq 0 \quad \text{and} \\ &\Big(-N(u,v,t) \cdot C^{\text{outward-1}}(s,t) \geq 0 \quad \text{or} \\ &\quad -N(u,v,t) \cdot C^{\text{outward-2}}(s,t) \geq 0\Big). \end{aligned} \tag{10}$$

Edge-to-Edge The *edge-to-edge* case involves two edge curves, $C_1(s_1, t)$ and $C_2(s_2, t)$. For this case, just a contact constraint is sufficient, given by the following system of three equations in three variables:

$$C_1(s_1,t) - C_2(s_2,t) = 0. \tag{11}$$

To define the incoming collision condition, we define two outward directions for each edge curve, as in the previous discussion for the edge-to-face case. The relative velocity between the edge curves must face in the same direction as at least one of the first curve's outward directions:

$$\begin{aligned} &(\frac{\partial C_1}{\partial t}(s_1,t) - \frac{\partial C_2}{\partial t}(s_2,t)) \cdot C_1^{\text{outward-1}}(s_1,t) \geq 0 \quad \text{or} \\ &(\frac{\partial C_1}{\partial t}(s_1,t) - \frac{\partial C_2}{\partial t}(s_2,t)) \cdot C_1^{\text{outward-2}}(s_1,t) \geq 0 \end{aligned} \tag{12}$$

Also, at least one of the outward directions on one curve must face away from one of the outward directions of the other curve. A logical combination of 6 inequalities is the result.

Vertex-to-Face, Vertex-to-Edge, Vertex-to-Vertex The *vertex-to-face* case involves a vertex $P(t)$ and a surface $S(u,v,t)$. As in the edge-to-edge case, a contact constraint is sufficient, of the form

$$P(t) - S(u,v,t) = 0. \tag{13}$$

where the point P is formed by evaluating a surface at a fixed point in its (u,v) parameter space, e.g.

$$P(t) \equiv S_a(u^{\text{fixed}}, v^{\text{fixed}}, t)$$

A system of three equations in three unknowns results. Similarly, a system of three equations in two unknowns results for the vertex-to-edge case, and a system of three equations in a single unknown for the vertex-to-vertex case.

The incoming collision condition can be derived by defining a number of outward directions for the colliding vertex, corresponding to the normal vector of each surface containing that vertex. The normal to the surface S must face away from at least one of these outward directions, as in the edge-to-face case. The relative velocity between the surface and the vertex must face in the same direction as the surface's normal, via

$$(\frac{\partial S}{\partial t}(u,v,t) - \frac{\partial P}{\partial t}(t)) \cdot N(u,v,t) \geq 0.$$

Similar systems of inequalities can be derived for situations where a vertex collides with an edge or another vertex.

B Inclusion Functions for Chebyshev Polynomials

Chebyshev polynomials are a good basis for a continuous representation of time behavior. They allow simple control of approximation error, and can be differentiated using a simple method to produce a Chebyshev representation of the derivative (see [PRES86, pages 158–165] for a discussion of the advantages of Chebyshev polynomials, their properties, and algorithms for their manipulation). The basis functions for a Chebyshev polynomials are

$$T_n(x) \equiv \cos(n \arccos(x))$$

which expand to a series of polynomials of the form

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_2(x) &= 2x^2 - 1 \\ &\vdots \\ T_{n+1}(x) &= 2xT_n(x) - T_{n-1}(x) \quad n \geq 1 \end{aligned}$$

The function $T_n(x)$ has $n+1$ extrema with values of ± 1 at the locations

$$x_i \equiv \cos(\frac{\pi i}{n}) \quad i = 0, 1, \ldots, n$$

The i-th extremum of the basis function T_n is either a minimum or maximum according to the rules

$$T_n(x_i) = \begin{cases} -1, & \text{if } (i+n) \equiv 1 \bmod 2 \\ +1, & \text{if } (i+n) \equiv 0 \bmod 2 \end{cases}$$

A Chebyshev approximation of order N is given by specifying N coefficients $c_i, i = 0, 1, \ldots, N-1$, which determine the polynomial

$$C(x) = \sum_{i=1}^{N-1} c_i T_i(x) + \frac{c_0}{2}$$

Given the order of the Chebyshev approximation function $C(x)$, N, we can easily compute an inclusion function for $C(x)$. Let the interval over which we are to bound $C(x)$ be given by $X = [x_0, x_1]$. As a preprocessing step, we first tabulate the locations of the extrema of the basis functions, up to some maximum order. (Note that the results can then be used for any approximating polynomial.) For each Chebyshev basis function, $T_i(x)$, $i = 0, 1, \ldots, N-1$, we first evaluate $T_i(x_0)$ and $T_i(x_1)$. We then determine whether any extrema of $T_i(x)$ occur in $[x_0, x_1]$ using the tabulated locations of the extrema. A lower bound on the basis function over $[x_0, x_1]$, b_i^0 is

$$b_i^0 \equiv \begin{cases} \min(T_i(x_0), T_i(x_1), -1), & \text{if min of } T_i(x) \in [x_0, x_1] \\ \min(T_i(x_0), T_i(x_1)), & \text{otherwise.} \end{cases}$$

Similarly, an upper bound is

$$b_i^1 \equiv \begin{cases} \max(T_i(x_0), T_i(x_1), 1), & \text{if max of } T_i(x) \in [x_0, x_1] \\ \max(T_i(x_0), T_i(x_1)), & \text{otherwise.} \end{cases}$$

The final inclusion function is then

$$\Box C(X) \equiv \sum_{i=1}^{N-1} c_i [b_i^0, b_i^1] + \frac{c_0}{2}$$

where operations are computed with interval arithmetic.

Scenes from test animations: In the following figure pairs, the upper image is the scene immediately before the collision, while the bottom image is the scene at the collision time. Points of contact are shown as white dots, which are uniformly distributed over regions where there are line and surface contacts. At the time of collision, surfaces become transparent to make the dots visible.

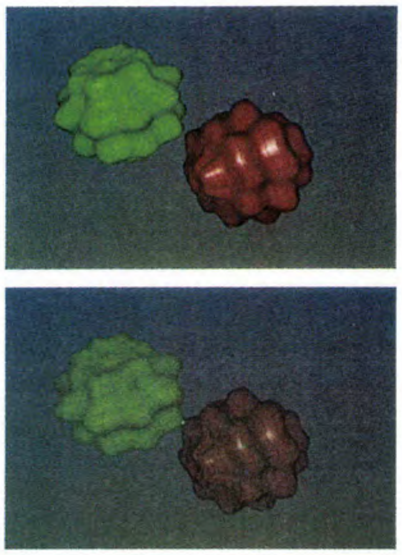


Figure 12: Two bumpy objects collide at one point.

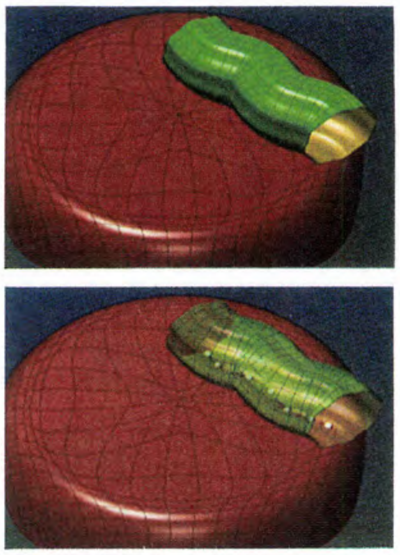


Figure 13: A time-varying tube contacts a cushion along a curve.

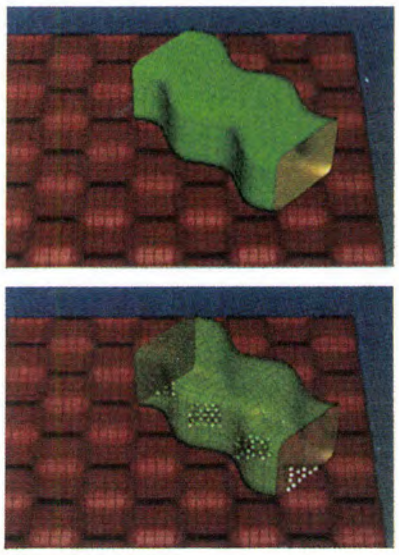


Figure 14: A wavy object contacts a raised checkerboard floor in several flat patches.

Scenes from "Fruit Tracing": This animation shows the results of collision detection for a more complicated setting involving hundreds of colliding objects. In this animation, moving parametric surfaces representing fruit are collided with a static lobster shape, defined as an implicit surface. (Lobster data generated by David Laidlaw, Matthew Avalos, Caltech, and Jose Jimenez, Huntington MRI Center.)

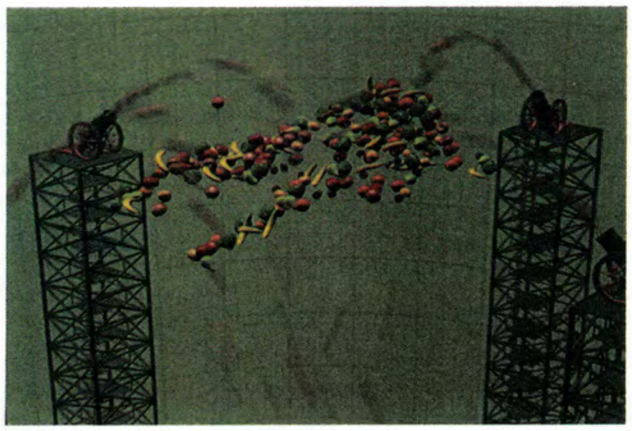


Figure 15: Colliding dynamic fruits.

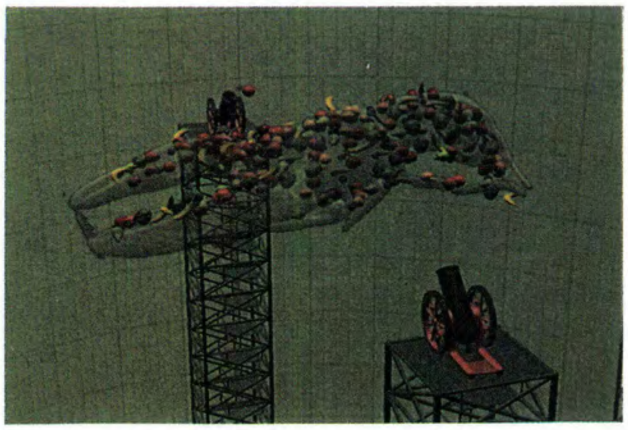


Figure 16: Scene showing lobster shape.

Sensor-Actuator Networks

Michiel van de Panne
*Eugene Fiume**

Department of Electrical Engineering and *Computer Science
University of Toronto†
Toronto, Canada, M5S 1A4

Abstract

Sensor-actuator networks (SANs) are a new approach for the physically-based animation of objects. The user supplies the configuration of a mechanical system that has been augmented with simple sensors and actuators. It is then possible to automatically discover many possible modes of locomotion for the given object. The SANs providing the control for these modes of locomotion are simple in structure and produce robust control. A SAN consists of a small non-linear network of weighted connections between sensors and actuators. A stochastic procedure for finding and then improving suitable SANs is given. Ten different creatures controlled by this method are presented.

CR Categories: G.3 [Probability and Statistics]: Probabilistic Algorithms; I.2.6 [Artificial Intelligence]: Learning, Robotics; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism - animation; I.6.3 [Simulation and Modeling] - Applications.

1.0 Introduction

Many recent investigations have involved the use of physical simulations for creating life-like animations of creatures. The problem is less one of simulation than it is of control: how should the muscles in the creature act to make it move in a desired way? Once known, the forces and torques produced by the muscles can be used as input to a physical simulation to obtain the resulting motion.

Broadly speaking, there exist two approaches to this problem. The first is to apply the laws of physics as constraints on the motion of the various parts of the creature while at the same time optimizing a given goal function. This goal function may be something like "get from point A to point B in the fastest way". An initial guess at the possible trajectory of the creature is iteratively refined until a trajectory is produced that both satisfies the physics constraints and optimizes the goal function.

The second approach is to synthesize a controller. A controller makes control decisions based upon sensory information presently available and does not explicitly calculate a trajectory. A controller thus makes use of feedback to perform its task. Our solutions, called sensor-actuator networks (SANs) are in the form of controllers. There are two main novel features of SAN controllers. First, they are entirely sensor-based. Control solutions are usually cast into a form where sensory information is processed to produce an estimate of the system state. For mechanical systems, the state is the tuple of values sufficient to specify the position and velocity of every point on the object. SANs have no notion of the state of the system as conventionally defined. Second, the stochastic synthesis procedures used to create SANs are unique and powerful tools.

A principal advantage of our method is that it requires less knowledge to use than other physically-based animation systems. The user provides the construction of a creature and can then ask the question "how would it move?" Our method can then yield several controllers providing feasible modes of locomotion (there is no *guarantee* that a suitable controller will be found). We have been surprised with many of the modes of locomotion discovered. Even the simplest objects are capable of a large repertoire of interesting motions. Many of the controllers generated using our method can take advantage of phenomena such as surface friction or collision impact, which can be simulated but are hard to incorporate in many optimization methods. Figure 1 shows modes of locomotion discovered for 4 different creatures.

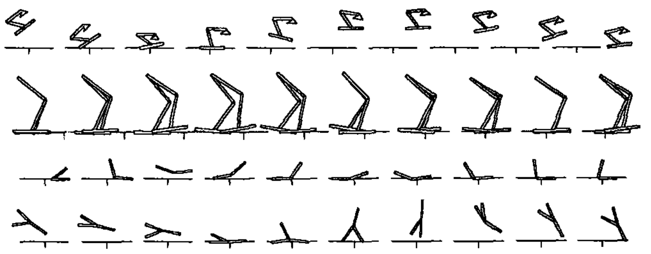


FIGURE 1. Some modes of locomotion using SANs

A sensor-actuator network is a non-linear network of weighted connections between a small number of binary sensors and the actuators (the muscles of our creatures). The network has internal delays, thereby giving it dynamic properties. In this paper we provide a method for determining the parameters associated with SANs in order to obtain instances of a desired behaviour. We show that SANs can be used to control a variety of interesting creatures.

The next section relates our work to important previous work. Section 3 discusses how to construct a creature. Section 4 presents the architecture of SANs. Section 5 gives the algorithms for synthesizing and fine-tuning SANs. Section 6 presents a variety of results, and section 7 concludes.

† Email: van@dgp.utoronto.ca, elf@dgp.utoronto.ca

The financial assistance of the Natural Sciences and Engineering Research Council of Canada, and of the Information Technology Research Centre of Ontario, is gratefully acknowledged.

2.0 Background

Developing control solutions for arbitrary dynamical systems is a difficult problem. Consequently, control solutions are usually specific to a constrained class of control problems. We briefly consider various classes of control problems applicable to animation and their solution techniques.

Linear systems are often the simplest to control. A linear system can be written in the form $\dot{x} = Ax + b$ where x is the state vector. Brotman and Netravali[5] applied linear control theory to the animation of some linear systems. Unfortunately, most interesting systems are very non-linear.

We shall define *smooth* systems as being those that are not necessarily linear, but whose state variables are C^0 continuous over time. This implies positions and velocities are C^1 and C^0 continuous, respectively. This notably excludes mechanical systems that undergo collisions, which instantaneously lose velocity. Smooth systems have states that follow a continuous trajectory through state-space. The solution to smooth control problems typically involves iteratively refining a state trajectory[6][17][24] or performing dynamic programming[8][22]). Solutions to smooth problems are often useful for animation. The take-off and aerial trajectories of many jumping motions fall within this class.

Statically stable systems are those that can be effectively controlled through kinematic means. This means that the motion of a creature can be halted in mid-action and then resumed without it falling over or otherwise collapsing. These can often be controlled by a cyclic motion passing through the body or legs. Realistic animations of snakes and worms[14] and a cockroach[13] have been performed using suitable controllers.

Specialized controllers have been constructed for many systems through careful analysis and simplification of the motions involved. Examples of interesting specialized controllers for walking, hopping, swinging, and juggling are given in [9][18][20][21]. The work in *Making Them Move*[1] presents a variety of specific solutions to control problems in animation.

We shall deal with a class of control problems in which the systems are non-linear, non-smooth, and not statically stable. A large number of crawling, jumping, hopping, flipping, and walking creatures fall in this category. Our sensor-actuator networks (SANs) prove to be particularly adept at controlling creatures in this class. It is difficult to try to perform an optimization that incorporates discontinuities. It should be noted, however, that it is not difficult to simulate many systems of this class.We shall use this to our advantage by performing repeated trials to determine a suitable controller. This means that a minimal knowledge of the physics of the mechanical system is required. We carry out our trials using physical simulations, although they could equally well be carried out directly on the real objects in principle.

Our work has similarities with the work of Wilhelms and Skinner[23] and Braitenberg[3]. Both discuss controllers constructed using weighted connections between sensors and actuators (or *effectors*), possibly through intermediate nodes of some kind. Braitenberg presents a series of thought experiments showing that such networks are capable of producing complex and seemingly intelligent behaviour. Wilhelms and Skinner allow the user to interactively construct the mapping between sensors and actuators. Their creature consists of a rigid body that can propel itself in three dimensions using a jet. Several different kinds of nodes are suggested for use in the connection network. Examples of attraction and avoidance behaviours can be constructed using their approach. Our work will show that SANs, which are similar networks, can be *automatically* synthesized and can be used to control a large variety of physically realizable creatures. Furthermore, we show it is only necessary to consider *one* kind of network node.

Our synthesis technique embodies a *generate-and-test* philosophy that can also be found in some other recent work. Ngo and Marks[16] use it to achieve similar goals to ours. Maes and Brooks[11] learn to coordinate leg motions for an insect robot. Sims uses the same notion to produce striking images[19]. A discussion of search-based control methodology is given in [15].

McGeer's work on passive walking[12] illustrates the phenomenon of mechanical systems reaching and maintaining stable limit cycles. Such *attractors* or limit cycles have also been studied in neural networks and have been conjectured as being the basis for behavioural action in nervous systems. In our work, the dynamical system consisting of the SAN and the creature has a propensity for such limit cycles, in which case it results in periodic motions, some of which are the useful gaits we are looking for.

SANs are similar in topology to many artificial neural networks (ANNs), but are different in several respects. Non-recurrent ANNs have no internal delays and thus provide a static input-output mapping. Furthermore, the synthesis method we employ does not use derivative-based learning methods.

SANs are loosely related to Brooks' work[4] on subsumption architectures for the control of mobile robots. The subsumption architecture is used to implement control at various levels in a control hierarchy. Beer presents a model of an insect nervous system that produces interesting behaviour[2].

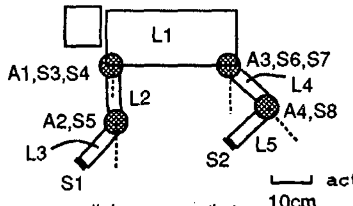
3.0 Constructing a Creature

The details of our method are best described through the use of an example. We shall use the *bounder* creature, shown in Figure 2. The bounder consists of 5 links, has 4 angular actuators, and 8 binary sensors. SANs take the binary sensor values as inputs and produce a set of desired angles or lengths as outputs. These desired values are used by proportional-derivative (PD) controllers to determine the internal torques or forces to apply. A PD controller is functionally equivalent to a spring and damper between the desired position and current position of a link. We shall now discuss each of the components of a creature in more detail and then discuss our simulator.

3.1 Mechanical Configuration

The user begins by specifying the desired mechanical configuration for the creature. All our creatures are built of rigid links and have planar dynamics, most of them operating in a vertical plane (i.e., under the influence of gravity). This is a limitation of our current implementation rather than the technique in general. As we shall

mechanical configuration:



link	mass (kg)
L1	1.0
L2	0.15
L3	0.10
L4	0.15
L5	0.10

sensors:

sensor	type	link	min	max
S1	touch			
S2	touch			
S3	angle	2	-180	-10
S4	angle	2	25	180
S5	angle	3	-180	-80
S6	angle	4	-180	42
S7	angle	4	55	180
S8	angle	5	-180	-105

actuators:

act.	min	max	ks	kd
A1	-20	30	0.4	0.01
A2	-85	-55	0.4	0.01
A3	40	55	0.4	0.01
A4	-110	-105	0.4	0.01

FIGURE 2. The bounder

soon see, planar creatures can have a very large repertoire of interesting behaviours. Specifying the physical structure of the creature requires specifying the mass, moment of inertia, and shape of each link, as well as how they are connected with joints.

3.2 Sensors

All sensors for our creatures are binary. If a sensor is 'on,' it produces a value of 1; otherwise it produces a 0. We currently use the four kinds of sensors shown in Figure 3. Touch sensors (e.g., S1 and S2 for the bounder) turn on when in contact with the ground and otherwise remain off. Angle sensors (e.g., S3-S8) determine if the angle of a limb is within the fixed range determined by the minimum and maximum angle specified for the sensor. This angle is measured relative to the link it is attached to, and the zero position is indicated with a dashed line in the creature diagrams. Eye sensors turn on if the *follow point* is in their cone of view. The follow point is a point to which some of our creatures will be attracted. One can thus control these creatures by dragging the follow point in front of them along a desired path. The cone of view for an eye is defined by a minimum and maximum angle. The zero-degree reference for eyes is shown in the creature diagrams with a dashed line. Length sensors are similar to angle sensors, but measure linear distances.

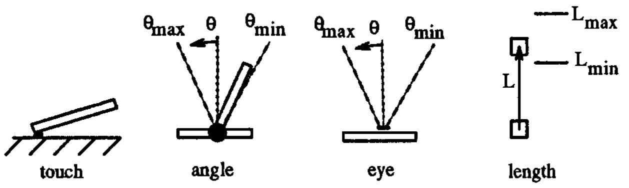

θ_{max} θ θ_{min}
θ_{max} θ θ_{min}
L_{max}
L_{min}
L
touch
angle
eye
length

FIGURE 3. Sensor types

Sensors are added to a creature after the mechanical configuration has been designed. It is necessary to use some intuition and creativity in deciding what sensors will be useful to a creature. Touch sensors are useful for points expected to be in contact with the ground. Angle sensors are useful for detecting when a limb has reached the end of its swing. Eye sensors are necessary to be able to track a follow point. Length sensors are useful in the control of linear actuators. The total number of sensors should also be considered. Too many or too few sensors can introduce problems for the network synthesis technique to be outlined shortly.

3.3 Actuators

We shall use two kind of actuators in our creatures: *linear* and *angular*. These are shown in Figure 4. Specifying an angular actuator requires specifying the joint concerned, the upper and lower limits that the desired angle can take (measured in degrees in our creature diagrams), and specifying the *strength* of the actuator.

The strength of an actuator is determined by the constants k_s and k_d associated with the PD controller for the joint. An angular actuator generates an internal joint torque $T = k_s(\theta_d - \theta) - k_d\dot{\theta}$, while a linear actuator generates a force $F = k_s(l_d - l) - k_d\dot{l}$. For many motions, choosing $k_d / k_s = 0.10$ provides suitable energetic motions with some damping. The value of k_s can often be chosen by doing a simple calculation. For example, a 10 degree deflection of limb L5 of the bounder should give a large enough force to support half the body weight when the leg is bent. A simple calculation will produce $k_s = 0.4$ Nm/deg, which is used for all the actuators in the bounder. No hard joint limits are provided by the actuators, although they can of course be implemented in the simulaton if desired.

Length actuators are similar to angular actuators, except that they exert linear forces between a pair of points. Besides the two points, the minimum and maximum desired length allowable for

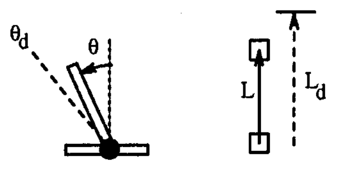


FIGURE 4. Actuator types

the actuator must be specified, as well as its strength. As with the angular actuators, the strength of the actuator is determined by its k_s and k_d constants.

3.4 Dynamics Simulation

Generating and using SANs requires only the capability to simulate the dynamics of a system. It is, however, beneficial to have a fast dynamics simulator, as our method for generating suitable SANs requires many simulation trials.

We make use of a dynamics compiler that uses the mechanical configuration information to generate a 'C' procedure which solves the equations of motion for a single time step. The procedure generates and solves a set of linear equations Ax = b, where x is the set of unknown accelerations. The elements in A and b are functions of the system state, the physical parameters of the system, the internal torques, and external forces. A recursive Newton-Euler formulation is used. This is $O(n^3)$ in the number of links and is quite suitable for $n < 10$.

The creatures are treated as free bodies in space. The external forces applied by the ground are calculated using stiff spring and dampers. We favour this approach as being simpler and more flexible than the alternative choice of reformulating the equations of motion upon impact of a link with the ground. The coefficient of friction of the ground can be set to a desired value. Other physical phenomena such as wind and water forces are easily added in as external forces. For example, for our fish creature, we calculate the water force for each link as $F = k\int(v \bullet N)\,dA$, where $v \bullet N$ defines the component of the velocity in the direction of the surface normal and A is the surface area of the link.

4.0 Sensor-Actuator Networks

SANs provide control by connecting sensors to actuators through a network of weighted connections. We will discuss in the next section the important question of how to determine the weight values. This section describes the structure and operation of SANs

A simplified example of a SAN is shown in Figure 5. The network consists of nodes and unidirectional weighted connections. The weights of the connections can take on values in a fixed range. In our implementation we choose integer values in the range [-2,2]. As shown, there are three kinds of nodes: sensor nodes, hidden nodes, and actuator nodes. The sensor nodes are fully connected to all hidden nodes and actuator nodes. All hidden nodes and actuator nodes are fully interconnected. The number of hidden nodes is usually chosen to be approximately equal to the number of sensor nodes. For example, the SANs for the bounder have 8 sensor nodes, 8 hidden nodes, and 4 actuator nodes.

Sensor nodes take on the values of their associated sensors. The hidden and actuator nodes function as shown in Figure 6. A node sums the weighted inputs and outputs a '1' if the sum is positive. This is a function similar to those performed in neural networks. It is important, however, for the controller to be a dynamical system on its own. This is effected by having a time delay associated with the operation of each node. This delay is implemented with the integrator and the following hysteresis function. The constants k_1

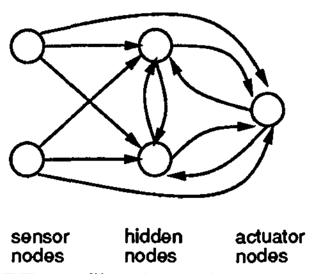


FIGURE 5. Topology of a sensor-actuator network

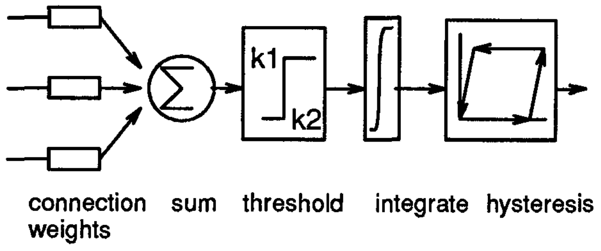


FIGURE 6. Function of a SAN node

```
sum = 0
for (each input i)
        sum = sum + input[i]*weight[i]
if (sum>0) then
        istate = istate + k1*dt
else
        istate = istate + k2*dt
if (istate>1.0) then istate = 1.0
if (istate<0.0) then istate = 0.0
if (output==1 and istate==0.0) then output=0
if (output==0 and istate==1.0) then output=1
```

FIGURE 7. Code corresponding to node function

and k_2 provide control over the time delay for a node to turn on and off, respectively. A hysteresis function is used instead of a simple threshold function to prevent the node output from chattering when some sensors make only momentary contact.

Actuator nodes make direct use of the result of the summing operation at a node to determine the desired angle or length of the actuator. The sum is first bounded to lie in a fixed range and is then linearly mapped to the length or angle range of the actuator. We bound the sum to the range [-2,2] in our implementation.

The function of a node is both simple to program and to build directly in hardware. The code implementing the functionality of a node is shown in Figure 7. The variable 'istate' is the internal state of a node associated with the integrator. We initialize istate to zero for all nodes at the start of a simulation. A hardware implementation would require three resistors, a capacitor, and three opamps per node.

The constants k_1 and k_2 should be chosen in relation to the expected duration of a periodic locomotion cycle for the creature. Thus the node delays for an elephant should be much larger than those for a mouse. The delays are given by $T_{on} = 1/k_1$ and $T_{off} = -1/k_2$ (note: $k_2 < 0$). Typically, good results can be obtained with $T_{on} = T_{off} = 0.25 \times T_c$, where T_c is the expected duration of one locomotion cycle. The delays used for all of our examples fall in the range of 0.07-0.15s. Because of the delays and the internal interconnectivity, the SAN has dynamic properties of its own that are important for generating useful control networks. We view the motions that the SANs produce as being the result of two interacting dynamical systems, that of the SAN and that of the creature itself. One can also think of SANs as being similar in function to the oscillators hypothesized to exist in some animals[13]. When the node delays were removed and the node functions were made more linear in several experiments, it most frequently resulted in a creature that rested in an immobile state after some initial motion. It is perhaps no coincidence that these are exactly the features that make the SANs operate as interesting oscillators by themselves.

5.0 Network Synthesis

We have now defined the construction of creatures and the structure of SANs, but we still need a method to obtain the weight values in the SAN. These will ultimately determine a creature's behaviour. Before outlining our solution, let us briefly consider another possible approach. Consider a top-down method whereby we assume we know the type of motion or desired gait and must now determine the details to generate it. Such an approach has several problems. First, it is often difficult to come up with the desired gait, especially if the creature has no counterpart in nature. Many of our creatures are capable of modes of locomotion that we never could have conceived of beforehand. Second, motion in the desired way might be physically impossible for the creature. We wish to restrict our search to strictly those solutions that are physically feasible. Lastly, the non-linear nature of the creature and its sensing devices makes any such top-down synthesis a very difficult task.

The top-down approach tries to determine a controller given a desired behaviour. The bottom-up approach that we shall take repeatedly generates and evaluates controllers (through simulations) until one or several are found that have desirable behaviours. We thereby exploit the fact that the simulation function is much less expensive than the top-down synthesis function. We have found that the SAN architecture defines a controller space that is highly populated with useful controllers.

Our network synthesis procedure consists of two phases. Phase 1 involves random SAN generation and evaluation. Phase 2 takes the best SAN controllers found in phase 1 and improves upon them. These two phases together define a coarse-to-fine strategy in searching for suitable controllers.

5.1 The Evaluation Metric

An evaluation metric is required for being able to determine the quality of a motion generated by any given SAN. For most of our creatures, this is simply the distance travelled in a fixed amount of time: $f_{eval} = |x(t_{final})|$. The creatures that travel the furthest are usually the ones that have an interesting mode of locomotion. We do usually not care whether the creature moves to the right or the left because both can yield interesting motions. The evaluation metric can be further qualified to be the distance travelled without falling over. For such events we set the evaluation metric to zero. To obtain a controller with energetic hops, the average height of the creature can be incorporated into the evaluation metric: $f_{eval} = \int_0^{t_{final}} (\dot{x} + ky^2)\, dt$. For creatures that should make use of their eye sensors to track a follow-point, the evaluation metric is: $f_{eval} = \int_0^{t_{final}} (v \bullet F)\, dt$, where v is the velocity of the creature and F the unit vector pointing towards the follow point. This metric is a simple measure of how well pursuit is maintained.

5.2 Phase 1: Random Generation and Evaluation

A controller is generated at random by choosing all of its weights at random. This is roughly equivalent to randomly selecting the reactions to be associated with each sensor. A randomly generated

SAN is then evaluated by simulating the behaviour of the creature with the SAN controller and calculating the evaluation metric.

The random search performed by phase 1 is used to discover different possible modes of locomotion for a creature. For this process to be an effective synthesis method, however, we must have some reasonable expectation of finding desirable controllers. At first glance, this does not seem a likely proposition. The space being randomly sampled is very large. For our bounder example it contains 5^{240} possible samples: each of the 240 weights can take any of 5 values. Furthermore, there is no guarantee that any point in this space corresponds to a useful controller.

It is our hypothesis that within the large search space there are many pockets containing useful weight combinations. While the SAN architecture clearly imposes a structure on the solutions, it is not obvious how to search this space. We have found that our dart-throwing strategy is successful a small but significant proportion of the time. The distribution of controllers ranked according to the evaluation metric is a sharply decreasing function with a long tail. Typically 1-5% of the random SANs result in useful motions. This has limitations; as creatures get more complex (6 or more links), fewer candidate motions are aesthetically pleasing.

5.3 Phase 2: Fine Tuning

A second fine-tuning phase may be applied to improve the controllers obtained as a result of the first phase. The fine-tuning phase makes small adjustments to some of the parameters of the SAN and the creature to improve its performance. The weights chosen in the previous step are not among the parameters to be adjusted, however. Because of the non-linear operation of the nodes in the SAN, small changes in the weights often result in either no change at all or a very large change in the dynamics of the system.

The parameters we shall adjust in the fine-tuning phase are those associated with the sensors and the actuators, as well as the delays for the hidden nodes. This list of parameters is shown in Figure 8. We wish to retain the same fundamental mode of locomotion but *evolve* the values of some of the parameters to obtain an improved gait. There are 4 parameters to adjust for each actuator, 2 for each network node, and 2 for each sensor. For our bounder, this results in a total of 44 adjustable parameters.

k1	delay in SAN node for turning on
k2	delay in SAN node for turning off
Amin	minimum desired angle or length for actuator
Amax	maximum desired angle or length for actuator
ks	spring constant for actuator
kd	damper constant for actuator
Smin	lower bound of sensing range for sensor
Smax	upper bound of sensing range for sensor

FIGURE 8. Adjustable parameters for SAN fine-tuning

The approach taken in phase 2 is to make small positive or negative perturbations to randomly chosen parameters. Evaluations are then performed to see if a given change improves the result. We consider two different approaches for performing and evaluating the parameter changes. These are stochastic gradient ascent (SGA) and simulated annealing (SA). These two approaches are distinguished in that one searches for a local optimum, while the second searches for a global optimum.

The structure of the SGA algorithm is shown in Figure 9. The simulated annealing algorithm used is equivalent to that described in [7] and [10]. The simulated annealing algorithm accepts some parameter changes that result in worse performance in order to be able to escape local minima (or maxima in our case). Both algorithms begin with the nominal parameter values assigned when the sensors and actuators were first designed.

```
for (1000 trials)
    randomly choose a parameter to vary
    perturb the parameter value by +delta or -delta
    evaluate the new creature by simulation
    if (creature improved) then
        keep change
    else
        reject change
```

FIGURE 9. Pseudocode for stochastic gradient ascent

A simulated annealing run of 1000 trials was usually sufficient to produce good results. The annealing schedule consisted of 65 evaluations at each annealing temperature, with the next 'temperature' being 0.75 times the previous one. In some cases SGA produced better results than SA for the same number of evaluations. For other cases the reverse was true. We expect that simulated annealing would find better solutions given many evaluations and a suitable annealing schedule, while stochastic gradient ascent quickly finds a reasonable local maximum.

6.0 Results

SANs appear to be capable of serving as useful controllers for almost any relatively simple creature. We have successfully experimented with a total of 10 creatures. These creatures are listed in Table 1, and some of their mechanical configurations are given in the Appendix. The creatures are drawn in the vertical plane and make use of the ground to propel themselves forward, with the exception of the fish. The fish makes use of the reaction forces of water to propel itself forward. The crawler consists of four point masses placed in a rectangular configuration. These masses experience less friction sliding forward than backwards. The crawler uses this property to be able to move forward.

Most of the creature designs are the original and only attempts at designing the creature. There remains, however, a certain measure of intuition involved in determining what sensors and actuators might be useful to the creature. The angle ranges of the sensors and actuators for both the bounder and luxo were varied in several experiments before settling on the given choices, which seem to be capable of yielding a broad range of interesting gaits. The forces and torques produced by the SAN controllers are usually not smooth because of the presence of impacts and the use of binary sensors.

TABLE 1. The experimental creatures

creature	links	sensors	actuators	hidden nodes	speed cm/sec
crawler	4	8	2	10	11
fish	4	6	2	5	19
bounder	5	8	4	8	115
luxo	3	6	2	6	79
cart	2	6	1	5	23
walker	6	11	5	6	101
twolink	2	6	1	5	12
threelink	3	8	2	7	33
fourlink	4	11	3	8	55
star	3	8	2	7	9

An evaluation of 200 random controllers (phase 1) finds useful controllers for each of the creatures, although a wider variety of gaits can be obtained by evaluating more. We retain the top 10 results automatically for examination by the user. For most creatures we use the distance moved in 6 seconds as an evaluation metric. A 30 second evaluation is used for the crawler and the fish in order to be able to test their tracking ability. An evaluation of 200 controllers thus requires 1200 seconds of simulation for most creatures. This can take from 1 hour for the cart creature to 6 hours for the walker creature (on a Sun SPARC IPC).

The crawler and the fish were designed to perform tracking. Their purpose in life is to always swim or crawl towards the follow point. They perceive this follow point only through their binary eye sensors. The joint connecting link L4 to L3 in the fish has a passive angular spring and damper. This joint and link L4 serve as the caudal fin of the fish, a necessary feature for efficient swimming. The crawler and the fish are creatures that form a dynamical system with their SANs that is capable of tracking an object successfully. They can do so with binary eyes and have never been given any information on how to move forward, turn, or associate eye information with turning. Figure 10 shows an example of the pursuit motion of the fish.

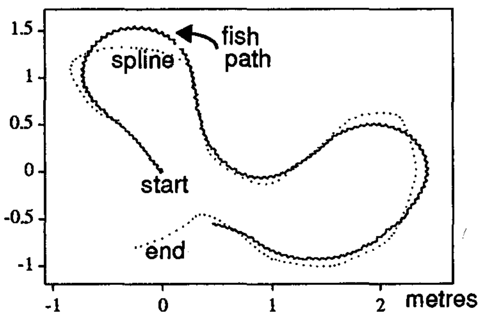


FIGURE 10. The fish chasing a point being dragged along a spline curve.

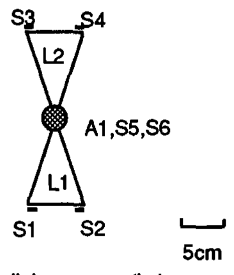
6.1 Variety of Solutions

The most novel modes of locomotion were discovered for very simple creatures, such as those having only two or three links. The cart creature, shown in Figure 11, is perhaps the best example. The creature derives its name from 'cartwheel' because it was originally designed with the hope it could move by performing cartwheels. Performing cartwheels is indeed one of the physically-feasible modes of locomotion discovered. The creature is also capable of four other unique and valid gaits, however. All 5 modes are shown in Figure 12. These modes were found in a run of 1000 controller evaluations. Several modes looked promising, but were unable to fully sustain periodic locomotion initially. These motions were easily fixed to yield proper periodic locomotion using the fine-tuning phase as described in the previous section.

A simple two-link chain is also capable of remarkable motions. It is important to note that one of the links is heavier than the other, which is necessary to avoid some situations where the chain is unable to locomote because of its symmetry. One SAN moves the creature forward using a flapping motion that lifts the joint vertex upwards and forwards in small hops. Another performs repeated 'jumps' onto its back. Yet another manages to get the links into a position such that it can do a big aerial jump. It 'falls' upon landing, but manages to get up to perform another jump, forming a repetitive motion.

More complex figures such as the bounder and the walker produce gaits more familiar to us. The bounder has gaits moving in small hops, big hops, shuffling, and others which are difficult to describe succinctly. The walker moves by performing shuffles, hops, and taking alternating steps. For many objects, the gaits

mechanical configuration:



sensors:

sensor	type	link	min	max
S1	touch	L1		
S2	touch	L1		
S3	touch	L2		
S4	touch	L2		
S5	angle	L2	-180	-120
S6	angle	L2	120	180

actuators:

act.	min	max	ks	kd
A1	-140	140	0.002	0.0001

link	mass (kg)
L1	0.10
L2	0.10

FIGURE 11. The cart creature

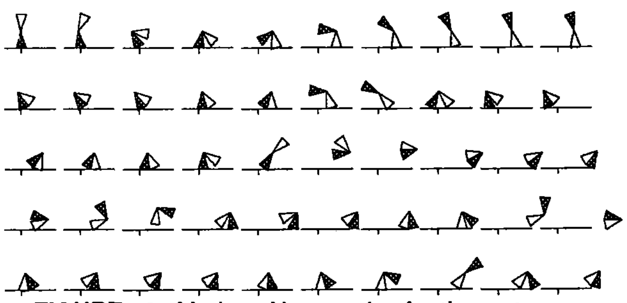


FIGURE 12. Modes of locomotion for the cart

produced are dependent upon the coefficient of friction. This is particularly the case for the three-link and four-link chain creatures.

Many of the solutions not ranking among the best in terms of the evaluation metric can also be interesting. Some creatures fall over but can still succeed in moving forward while on their back. Others move a bit and then become fixed, apart from a repeated twitching motion of leg. This is also the case for creatures that fall over into a state from which they cannot get up. These often repetitively twitch their legs in a style reminiscent of an upside-down insect.

There are limits to the complexity of motion that can be expected to emerge by evaluating randomly-generated controllers. It would be futile, for example, to expect to use our method directly to find a SAN to control a reasonable model of a human body in performing a high-jump. As we will discuss shortly, however, we believe that more complex controllers might be achievable through an evolutionary approach.

6.2 Robustness of Solutions

SANs are also an interesting control structure because they can provide robust control. Figure 13 shows a bounder moving over rough terrain using a SAN that was chosen for its performance over flat terrain. A measure of robustness could be included in the search procedure of phase 1 by performing all the evaluations over terrain of the desired roughness. We believe the robustness is a property related to the inherent simplicity of SAN controllers.

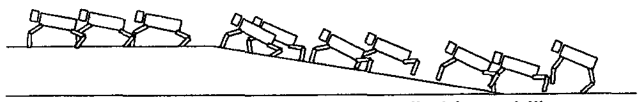


FIGURE 13. The bounder creature climbing a hill

The robustness has limits because the creatures as presently constructed have no means of detecting the upcoming terrain. As such, they are functioning in a manner equivalent to a person walking over rough terrain in the dark. When moving over rough terrain, the same mode of locomotion is usually maintained, with the variations in terrain causing the timing of events to be changed slightly. The gaits are not as robust once they have been fine-tuned. The situation is analogous to running blindly as opposed to walking blindly. Motions optimized for speed are usually less stable.

6.3 Evolution

The second phase of the network synthesis involves making small changes to some parameters of the sensors, actuators, and delays in the SAN. One could also consider including small changes to the actual physical parameters of the system in order to improve its motion. These parameters could include the link lengths, masses, and points of attachment. Such changes are akin to an artificial kind of evolution. For adherance to the constraints of biological systems, the values of these parameters should be interrelated. For example, a stronger spring constant in the actuator, corresponding to a stronger 'muscle', should increase the mass of the appropriate link. Similarly, a larger link should also have an increased mass.

The evolutionary principle could also perhaps be applied to the synthesis of controllers for more complex systems. It is unlikely that random search will stumble upon the best mode of locomotion for a complex articulated figure, such as a good model of a horse. We feel that this is not a large shortcoming of our method because nature herself does not directly arrive at suitable controllers for such creatures. The skeleton, musculature, and control for a horse are the result of a long series of evolutionary changes. We believe it might be possible to arrive at complex controllers by first beginning with the control of simpler figures, such as those that have fewer joints or that have more stable locomotion.

7.0 Conclusions

We have presented a new method that automatically searches for interesting and physically-feasible modes of locomotion for arbitrarily-designed creatures. The solutions are in the form of simple controllers that use binary sensors as input and produce actuator forces and torques as output. Sensor-actuator networks are a compact representation of the complex and varied gaits that they produce. A typical controller has 240 weights that can take on 5 values, and can thus be represented in 70 bytes.

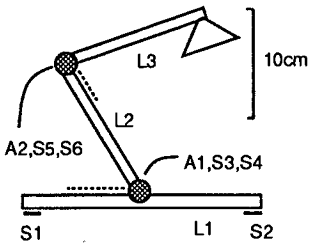
There are several advantages to using SANs. Using the method requires no knowledge of the underlying equations of motion. The user can rapidly construct a new creature and have the method 'discover' several ways it can locomote. It is easy to generate controllers that take advantage of more complex physical phenomena. As an example, the coefficient of friction can have a profound impact on the choice of the best gait. SANs can control mechanical systems that do not always have smooth motion. The user can influence the motion through the specification of the evaluation metric and evaluation terrain in order to achieve the desired speed, energy, height, and robustness of the control. The ability to fine-tune the sensors, actuators, and delays in the SAN means the creature designer need not choose the optimal design parameters to begin with.

SANs have some disadvantages from the point of view of an animator. They do not at present provide the user with as much control over the resulting motion as keyframing or physics-as-constraints methods. Furthermore, there is no guarantee that any useful solution will be found. SANs in their current incarnation do not work very well for systems dominated by linear or smooth dynamics. The synthesis method also fails to find the best modes of locomotion for complex creatures. Our method always results in physically-realizable motions, something that can be an advantage or a restriction in animation.

We believe there are many possible future directions for this work. Many experiments can be carried out using other types of architectures, sensors, and actuators. The fine-tuning phase could be used to determine how to insert passive elastic elements in the system to reduce the amount of expended energy for a given mode of locomotion. The use of SANs as the bottom level of a control hierarchy needs to be investigated. More understanding and analysis of the dynamical systems formed by the SANs and the creatures is needed.

Appendix

mechanical configuration:



link	mass (kg)
L1	0.05
L2	0.10
L3	0.30

sensors:

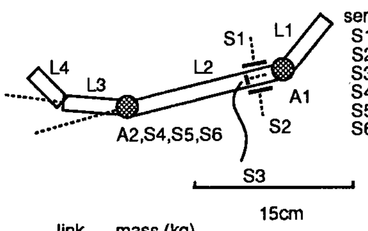
sensor	type	link	min	max
S1	touch	L1		
S2	touch	L1		
S3	angle	L2	−180	−70
S4	angle	L2	−53	180
S5	angle	L3	−100	75
S6	angle	L3	140	180

actuators:

act.	min	max	ks	kd
A1	−70	−50	0.05	0.001
A2	60	150	0.04	0.001

Luxo

mechanical configuration:



link	mass (kg)
L1	0.2
L2	1.0
L3	0.3
L4	0.1

sensors:

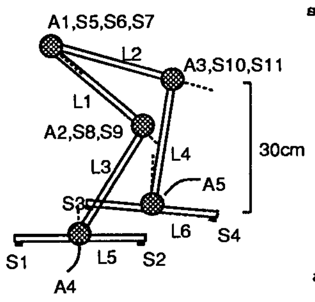
sensor	type	link	min	max
S1	eye	L2	10	180
S2	eye	L2	−10	10
S3	eye	L2	−180	−10
S4	angle	L3	−180	−25
S5	angle	L3	−25	25
S6	angle	L3	25	180

actuators:

act.	min	max	ks	kd
A1	−20	20	0.003	0.001
A2	−40	40	0.006	0.001

The fish

mechanical configuration:



link	mass (kg)
L1	0.3
L2	0.3
L3	0.2
L4	0.2
L5	0.1
L6	0.1

sensors:

sensor	type	link	min	max
S1	touch	L5		
S2	touch	L5		
S3	touch	L6		
S4	touch	L6		
S5	angle	L2	−180	−15
S6	angle	L2	−15	−5
S7	angle	L2	15	180
S8	angle	L3	−180	−85
S9	angle	L3	−75	180
S10	angle	L4	−180	−85
S11	angle	L4	−75	180

actuators:

act.	min	max	ks	kd
A1	−30	30	0.15	0.01
A2	−105	−70	0.15	0.01
A3	−105	−70	0.15	0.01
A4	20	30	0.10	0.00
A5	20	30	0.10	0.00

The walker

References

[1] N. Badler, B. Barsky, and D. Zeltzer (Eds). *Making Them Move*. Morgan Kaufmann, 1991.

[2] R. Beer. *Intelligence as Adaptive Behavior*. Academic Press, 1990.

[3] V. Braitenberg. *Vehicles: experiments in synthetic psychology*. MIT Press, 1984.

[4] R. A. Brooks. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation*, 2, 1 (March 1986), 14-23.

[5] L. S. Brotman and A. N. Netravali. Motion Interpolation by Optimal Control. Proceedings of SIGGRAPH '88. In *ACM Computer Graphics*, 22, 4 (August 1988), 309-315.

[6] M. F. Cohen. Interactive Spacetime Control for Animation. Proceedings of SIGGRAPH '92. In *ACM Computer Graphics*, 26, 2 (July 1992), 293-302.

[7] E. Fiume and M. Ouellette. On distributed, probabilistic algorithms for computer graphics. *Proceedings of Graphics Interface '89*, 211-218, 1989.

[8] M. Girard. Constrained Optimization of Articulated Animal Movement in Computer Animation. In *Making Them Move*, Morgan Kaufmann, 1991, 209-232.

[9] J. K. Hodgins, P. K. Sweeney, and D. G. Lawrence. Generating Natural-looking Motion for Computer Animation. *Proc. of Graphics Interface '92*, 265-272.

[10] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220,13 (May 1983), 671-680.

[11] P. Maes, R. Brooks. Learning to Coordinate Behaviors. *Proc. of AAAI '90*, 1990, 796-802.

[12] T. McGeer. Passive Walking with Knees. *Proceedings of the IEEE International Conference on Robotics and Automoation*, 1640-1645, 1990.

[13] M. McKenna and D. Zeltzer. Dynamic Simulation of Autonomous Legged Locomotion. Proceedings of SIGGRAPH '90. In *ACM Computer Graphics*, 24, 4 (August 1990), 29-38.

[14] G. S. P. Miller. The Motion Dynamics of Snakes and Worms. Proceedings of SIGGRAPH '88. In *ACM Computer Graphics*, 22, 4 (August 1988), 169-178.

[15] W.-Y. Ng. Perspectives on Search-Based Computer-Aided Control System Design. *IEEE Control Systems Magazine*, 13, 2 (April 1993), 65-72.

[16] J. T. Ngo and J. Marks. Spacetime Constraints Revisited. Proceedings of SIGGRAPH '93. In *ACM Computer Graphics*, 27 (August 1993).

[17] M. G. Pandy, F. E. Zajac, E. Sim, and W. S. Levine. An Optimal Control Model for Maximum-Height Human Jumping. *J. Biomechanics*, 23, 12, 1185-1198, 1990.

[18] M. H. Raibert and J. K. Hodgins. Animation of Dynamic Legged Locomotion. Proceedings of SIGGRAPH '91. In *ACM Computer Graphics*, 25, 4 (July 1991), 349-358.

[19] K. Sims. Artificial Evolution for Computer Graphics. Proceedings of SIGGRAPH '91. In *ACM Computer Graphics*, 25, 4 (July 1991), 319-328.

[20] A. J. Stewart and J. F. Cremer. Beyond Keyframing: An Algorithmic Approach to Animation. *Proceedings of Graphics Interface '92*, 273-281, 1992.

[21] M. van de Panne, E. Fiume, and Z. G. Vranesic. A Controller for the Dynamic Walk of a Biped Across Variable Terrain. *Proceedings of the 31st IEEE Conference on Decision and Control*, 1992.

[22] M. van de Panne, E. Fiume, and Z. G. Vranesic. Reusable Motion Synthesis Using State-Space Controller. Proceedings of SIGGRAPH '90. In *ACM Computer Graphics*, 24, 4 (August 1990), 225-234.

[23] J. Wilhelms and R. Skinner. An Interactive Approach to Behavioral Control. *Proceedings of Graphics Interface '89*, 1-8, 1989.

[24] A. Witkin and M. Kass. Spacetime Constraints. Proceedings of SIGGRAPH '88. In *ACM Computer Graphics*, 22, 4 (August 1988), 159-168.

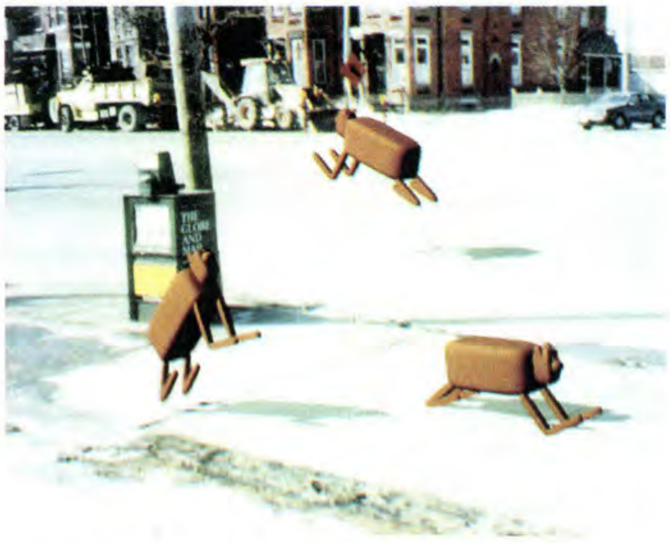


FIGURE 14. Bounders on the run

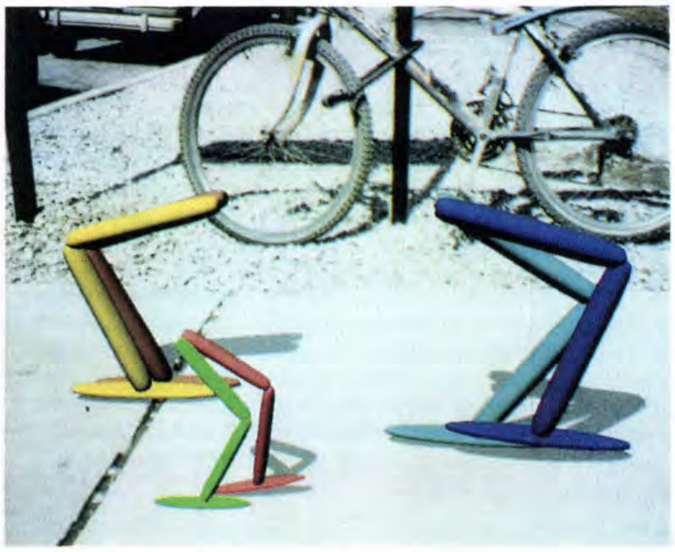


FIGURE 15. Walking on campus

Spacetime Constraints Revisited

J. Thomas Ngo
Graduate Biophysics Program
Harvard University[*]

Joe Marks
Cambridge Research Lab
Digital Equipment Corporation[†]

Abstract

The Spacetime Constraints (SC) paradigm, whereby the animator specifies what an animated figure should do but not how to do it, is a very appealing approach to animation. However, the algorithms available for realizing the SC approach are limited. Current techniques are local in nature: they all use some kind of perturbational analysis to refine an initial trajectory. We propose a global search algorithm that is capable of generating multiple novel trajectories for SC problems from scratch. The key elements of our search strategy are a method for encoding trajectories as behaviors, and a genetic search algorithm for choosing behavior parameters that is currently implemented on a massively parallel computer. We describe the algorithm and show computed solutions to SC problems for 2D articulated figures.

CR Categories: I.2.6 **[Artificial Intelligence]**: Learning—*parameter learning*. I.2.6 **[Artificial Intelligence]**: Problem Solving, Control Methods and Search—*heuristic methods*. I.3.7 **[Computer Graphics]**: Three-Dimensional Graphics and Realism—*animation*. I.6.3 **[Simulation and Modeling]**: Applications.

Additional Key Words: Genetic algorithms, evolutionary computation, massive parallelism.

1 Introduction

The Spacetime Constraints (SC) formulation was proposed in 1988 by Witkin and Kass as a powerful paradigm for indirectly controlling the physically realistic motion of articulated figures ("creatures") for animation purposes [17]. The essence of the SC approach is to have the animator specify what the animated creature should do, and the computer determine how the creature should do it. Specifically, the animator defines:

- the physical structure of the creature;
- the actuators that control the creature's internal configuration; and
- criteria for evaluating the creature's motion.

[*]Present address: Interval Research Corporation, 1801-C Page Mill Road, Palo Alto, CA 94304. Email: ngo@interval.com.
[†]One Kendall Square, Building 700, Cambridge, MA 02139. Email: marks@crl.dec.com.

The computer is left with the task of computing a physically realistic trajectory for the creature that is optimal according to the animator-supplied criteria.

The computer's role in this paradigm is quite challenging.[1] Finding globally optimal solutions to SC problems is hard for two main reasons [13]:

- Multimodality—even when the problem is suitably discretized, there are an exponential number of possible trajectories that a creature can follow, many of which may be locally optimal or near optimal.
- Search-space discontinuities—a small change in the behavior of a creature's actuators can lead to a large change in its trajectory.

Current algorithmic techniques for SC problems sidestep the difficulties of global optimization entirely by settling for some form of local optimization: typically an initial trajectory is perturbed in some way until it is locally optimal [2, 17, 3]. In this paper we propose a global search algorithm that fulfills a complementary need by generating multiple near-optimal trajectories for SC problems from scratch. The key elements of our search strategy are a method for encoding trajectories as sets of stimulus-response behavior rules [14, 16, 13], and a genetic algorithm (GA) [10, 8, 5] for choosing behavior parameters that is currently implemented on a massively parallel SIMD computer.

2 Algorithm

Our algorithm can be described in summary as follows:

- A dynamics module (§2.1) simulates a physically correct virtual environment in which the effects of creature behaviors may be tested by trial and error.
- A behavior module (§2.2) generates such behaviors using a parameterized algorithm that is based on the concepts of stimulus and response [14].
- A search module (§2.3) uses a genetic algorithm to choose values for the stimulus and response parameters that will generate near-optimal behaviors according to the evaluation criteria for the given SC problem.

[1]This is not to suggest that the animator's role has been trivialized completely—far from it! Specifying appropriate evaluation criteria for a creature's motion is a challenging problem, as is the task of reducing a complex animation problem to a concatenated sequence of related SC problems [3].

2.1 Physical simulator

Like Hahn's [9], our physical simulator employs forward *dynamics*, but treats an articulated figure as an autonomously deforming object without internal degrees of freedom. The deformations are produced *kinematically* by the creature's stimulus-response control algorithm (§2.2). The principal advantage of this approach is dramatically reduced CPU cost.[2] We find that in practice there is little sacrifice in terms of physical realism even though torques about the joints are never computed explicitly. If torque limits were to be important for a particular application, a penalty for excessive torques could be added to the evaluation criteria; the torques can be computed using inverse dynamics [11]. However, we have found that simply limiting the rate at which a creature's internal degrees of freedom can accelerate—something that can be specified trivially in our behavior-based representation—is enough to produce visually reasonable behavior.

Friction is taken to be static when one joint is in contact with the floor, and slippage is proportional to contact force when two joints touch. Collision and contact forces are treated essentially[3] as in Baraff [1]. We specialized Baraff's elegant treatment to two dimensions, then extended it to accommodate autonomous deformations. For a rigid polygon in 2D interacting with a flat 1D floor, the contact-force computation is much simpler than in the 3D case, because the number of contact points is limited to two. The extension to accommodate autonomous deformations is straightforward as well. The key to Baraff's technique is that the relative normal acceleration ($\ddot{\chi}_i$) for a given contact point i turns out to be a linear function of the vector of normal contact forces ($\vec{f}$). The relation remains linear after adding a contribution to $\ddot{\chi}_i$ that takes into account the creature's deformation, so the procedure for satisfying the "non-penetration" and "non-stickiness" constraints ($\ddot{\chi}_i \geq 0$ and $f_i \geq 0$) remains essentially unchanged.

2.2 Stimulus-response representation

Trial behaviors to be tested in the simulator (§2.1) are generated by the stimulus-response (SR) control algorithm. This algorithm neither learns nor plans; rather, it causes the creature to execute instinctive reflexes triggered by conditions sensed in its virtual environment. The conditions are called *stimulus functions* and the reflexes are called *responses*; and their parameters remain fixed for the duration of a trial behavior. The search module (§2.3) finds values for both the stimulus parameters and the response parameters essentially by trial and error. The use of these SR parameters in lieu of a more conventional time series of configurations or forces may be the most important factor in the success of our approach to SC problems.

A *response* is a prescription for changing the creature's shape smoothly. It consists of a time constant and a complete set of target values for the creature's internal angles. Applying the response for one time step means iterating through one time step of the critically damped equation of motion

$$\tau^2 \ddot{\theta}_i + 2\tau \dot{\theta}_i + (\theta_i - \theta_i^0) = 0$$

[2]Physical simulation is the CPU-intensive portion of our approach.

[3]A modification was necessitated by the SIMD architecture of our machine, because of which we implemented a fixed-timestep integrator with analytic recomputation after a mid-timeslice collision, rather than a more accurate variable-timestep integrator with instantaneous handling of the impulsive forces that arise from collisions.

for each of the internal angles, where τ is the time constant, θ_i^0 is the target value of the internal angle, and θ_i is its actual value. The effect of this equation of motion is to cause the real shape of the creature, given by $\{\theta_1, \theta_2, \ldots, \theta_N\}$, to approach the target shape $\{\theta_1^0, \theta_2^0, \ldots, \theta_N^0\}$ smoothly, even when the target shape changes abruptly due to a switch from one response to another.

To define a stimulus function we must introduce the concept of a sense variable. A *sense variable* is some real-valued function of the physical environment. Our standard list includes:

- proprioceptive senses—each joint angle;
- tactile senses—the force exerted by each rod endpoint on the floor (and *vice versa*);
- kinesthetic sense—the vertical velocity of the center of mass; and
- position sense—the vertical position of the center of mass relative to the floor.

A *stimulus function* is a scalar function defined over sense space. If the sense variables are $\{v_1, v_2, \ldots, v_V\}$, then the expression that we use for a stimulus function is:

$$W\left\{1 - \max_{j=1}^{V} \left[\lambda_j (v_j - v_j^0)\right]^2\right\},$$

where the v_j^0 and λ_j are parameters determined by the search module (§2.3), the weight W is:

$$W = \sum_{j=1}^{V} \log\left(\frac{\lambda_j}{\lambda_j^{\min}}\right),$$

and $\lambda_j^{\min}$ is the predetermined smallest permissible value of λ_j (§2.3). (In practice all of these constants are normalized so that the sense variables v_j fall between 0 and 1.) The locus of points in sense space for which the stimulus-function expression is positive—its *sensitive region*—is a hyper-rectangle with dimensions $\{2/\lambda_1, 2/\lambda_2, \ldots, 2/\lambda_V\}$ centered at $\{v_1^0, v_2^0, \ldots, v_V^0\}$.

The set of SR parameters for a creature consists of an array of SR pairs (10 pairs in all the tests we have run to date). In the following pseudocode, which describes how an SR array is used to generate behavior, the state variables are the creature's physical state and a pointer to the active response:

```
*Initialize* creature state from SC problem description
*Activate* response 0
**for** t = 1 **to** T
    *Determine deformation* for time t from active response
    *Simulate* resulting *dynamics* for time t
    *Measure* sense variables from the environment
    *Identify* highest-valued stimulus function
    *Activate* corresponding response if stimulus positive
**end for**
```

The final step may or may not change which response is active; a change is made only if the highest-valued stimulus function is positive. Thus, a response is typically active for several consecutive time steps and produces coherent motion.

2.3 Genetic algorithm

In §2.1 and §2.2 we described what a creature would do given a structural specification, initial conditions, and values for its stimulus and response parameters. In §3 we will supply examples of how the resulting behaviors might be evaluated quantitatively. Taken together, these ingredients define a scalar function of the SR parameters. It remains to show how to find near-optimal values of this function.

Our search procedure is a parallel genetic algorithm (GA) written in C* on a Thinking Machines CM-2 with 4096 processors. In a GA, a population of candidate solutions is subjected to a procedure that simulates biological evolution. Each candidate solution—each *genome*, in GA parlance—has some probability of being mutated, recombining with another genome, and dying, based on its value. In our implementation, each processor is responsible for evaluating one genome per generation. Our GA is described below in pseudocode:

```
do parallel
    Randomize genome
end do
for generation = 1 to number_of_generations
    do parallel
        Evaluate genome
        Select mate from another processor
        Cross genome with mate
        Mutate genome
    end do
end for
```

In this section we shall explain the component operations of this genetic algorithm: *Randomize*, *Select mate*, *Cross*, and *Mutate*. Although these operations have worked without modification for all of the test cases shown here and elsewhere [13], they should not be considered optimal.

Symbol	Description	Lo	Hi	Distribution
τ	Time constant	$2\Delta t$	$\frac{1}{4}T$	Logarithmic
θ_i^0	Target angles	$\theta_i^{\min}$	$\theta_i^{\max}$	Uniform
v_j^0	Stimulus centers	-0.5	1.5	Uniform
$2/\lambda_j$	Stimulus extents	0.4	4	Logarithmic

Table 1: The parameters that comprise one SR pair in the genome, their typical ranges, and their probability distributions in the initial random population. The timestep size and total simulation time are Δt and T, respectively; and the quantities $\theta_i^{\min}$ and $\theta_i^{\max}$ are lower and upper bounds on the joint angles taken from the structural specification of the creature.

Randomization Values for all of the parameters in the genome are chosen at random in the initial population, with the probability distributions given in Table 1. We then employ a form of hill-climbing to enrich the initial gene pool: after evaluating this initial population, we mutate and re-evaluate each solution four times, and on each processor choose the best out of five. Because of our highly specialized mutation operator (see below), this has the effect of producing a fairly non-random population that is skewed in favor of multi-step behaviors.

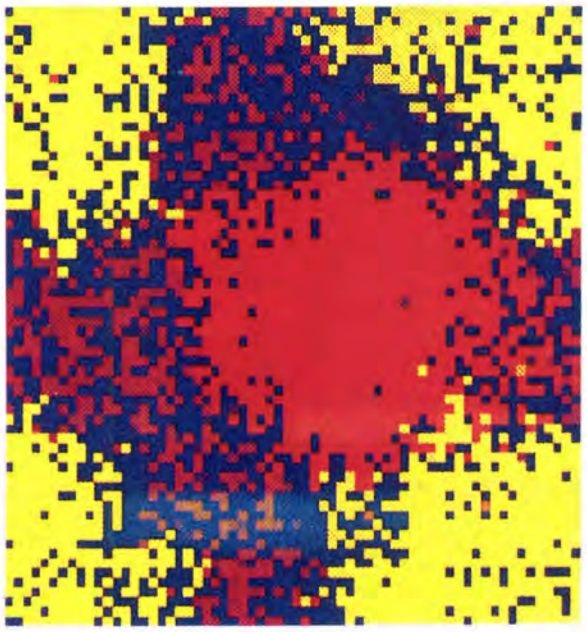


Figure 1: Visualization of a GA population. The display consists of 64×64 pixels, each one corresponding to a processor in the CM-2. Each pixel is colored according to the value of the solution stored at the corresponding processor: the colors range from dark blue for the worst solutions to bright yellow for the best ones. In this mature population two relatively homogeneous regions ("colonies") have emerged from a background of mediocre solutions; the corner areas form one region on the toroidal grid.

Mate selection This is the only step in the algorithm that requires interprocessor communication, and it is also the only stage at which better solutions are given an advantage. In the simplest GA's, mating between any two individuals is possible. To prolong the diversity of the population—this helps cope with multimodality—it is common practice to impose a geographic distribution on the population and to permit only local mating. In our implementation, the processors are laid out on an imaginary 64×64 toroidal grid; this scheme is especially easy to achieve on the CM-2. To choose a mate, each processor performs a random walk of (typically) 10 steps on the grid, and identifies the best solution encountered. If the best solution encountered is itself, it does not mate. (Thus, a solution that is best in its neighborhood is protected from crossover and mutation; this is a form of *elitism* [7].) Otherwise, it fetches a copy of the selected mate's genome in preparation for crossover.

This local mating scheme causes good "genes" (solution fragments) to diffuse slowly through the population, leading to the formation of "colonies" of similar solutions. Figure 1 depicts a population that arose in one of our experiments. As a colony spreads, more processors are put to work on variations of the supe-

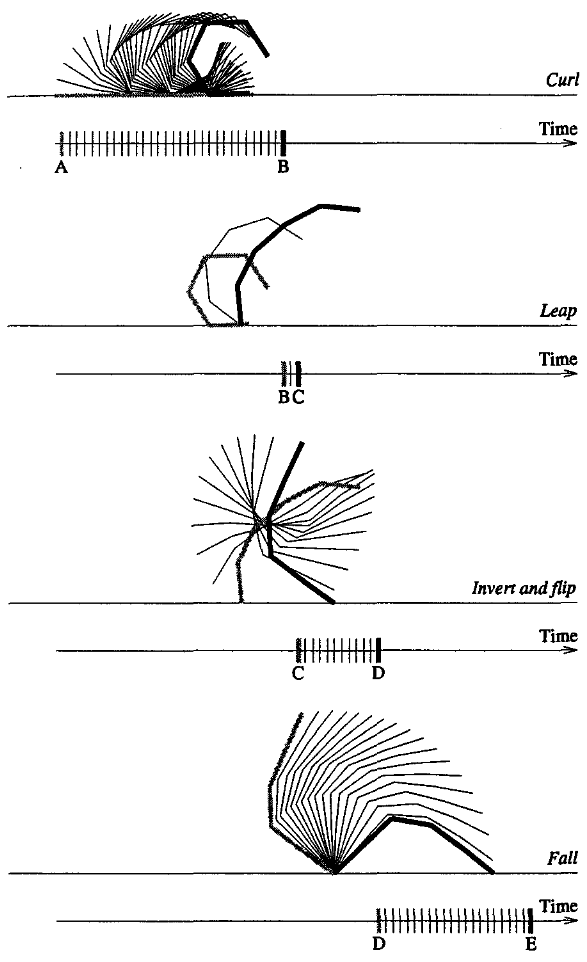


Figure 2: Five-Rod Fred moves to the right by curling and leaping. Since his initial configuration is symmetric, his curling action must be asymmetric in order to generate net rightward motion. Momentum is essential in permitting a forward leap. Inversion in the air is necessary to delay landing long enough to fall forward instead of backward. **Note:** In this and subsequent figures we have arbitrarily broken the motion into easily perceived phases. This is only for presentation purposes, and is not indicative of any aspect of the algorithm. Ticks along the horizontal time axis show which time slices are drawn, and in what linestyles. The first, intervening, and last frames in each phase are drawn with thick grey, thin black, and thick black lines, respectively. An italicized phrase at the lower right of each figure describes the corresponding action.

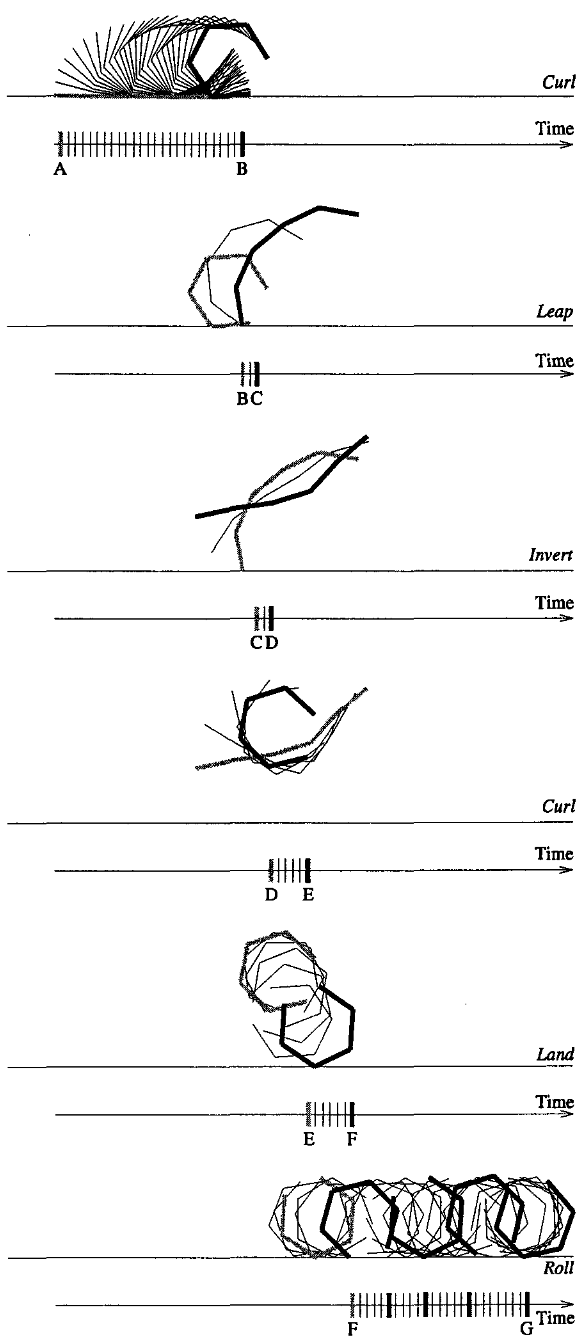


Figure 3: Fred rolls on landing. Because his curled configuration is a hexagon and not a circle, Fred bounces as he rolls.

rior solution that spawned the colony. But because the spreading is not instantaneous, suboptimal solutions still get some processing time. When one colony dominates (a typical outcome when genetic algorithms of this type are left to run indefinitely), the population is said to have converged.

Crossover The crossover operators used in the GA as it was originally described [10] are based on a relatively literal interpretation of the biological metaphor: the bit representations of the parameters in the genomes are placed end-to-end in linear fashion, and crossover operations call for snipping two genomes at analogous locations along the bit string, and swapping one pair of ends.

In the SR representation, a linear layout would not be very meaningful, and we know in advance that certain groups of parameters should migrate together between genomes. Therefore our crossover operator is tailored to treat the genome in a more structured fashion. Given the probabilities that we use, a hybrid might be constructed from two genomes (of ten SR pairs each) as follows: six intact SR pairs are taken from the mate and two from

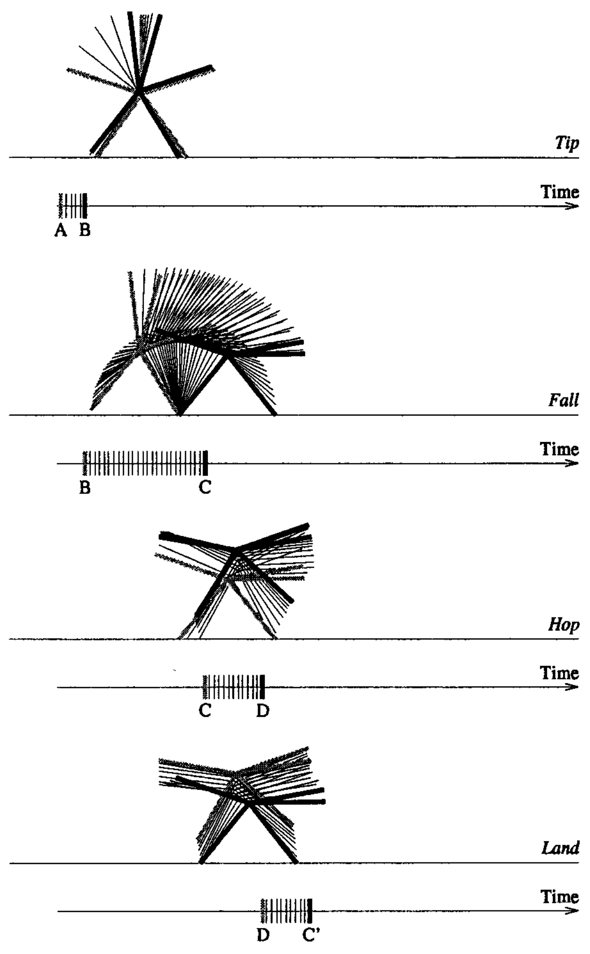


Figure 4: Mr. Star-Man canters on two side limbs. The C–D–C' sequence is repeated cyclically. Were Mr. Star-Man to have human-like strength and structure he would not be able to walk this way for very long, since his arm would get tired. This is one situation in which penalties for excessive torques might influence the range of behaviors discovered.

the original genome; one new SR pair is constructed by taking a stimulus and a response from each parent, respectively; and one new SR pair is constructed by uniform crossover [15], *i.e.*, each number in the SR data structure is copied at random from one or the other parent. The precise probabilities that we use in performing the crossover are not critical; we have not tried to optimize them because the values we use currently are probably irrelevant for future implementations of our general approach (§4). The key point is that the crossover operation must be tailored to the problem to get good performance.

Mutation Our mutation operator [13] is also tailored specifically for the SR representation. One SR pair is subjected to *creep* [5]; *i.e.*, each of the parameters in that SR pair is changed by a small amount. Another SR pair is randomized from scratch, with one stipulation that turns out to be quite important in coping with a large number of sense variables: that at least one corner of the sensitive region (§2.2) of the newly generated stimulus function

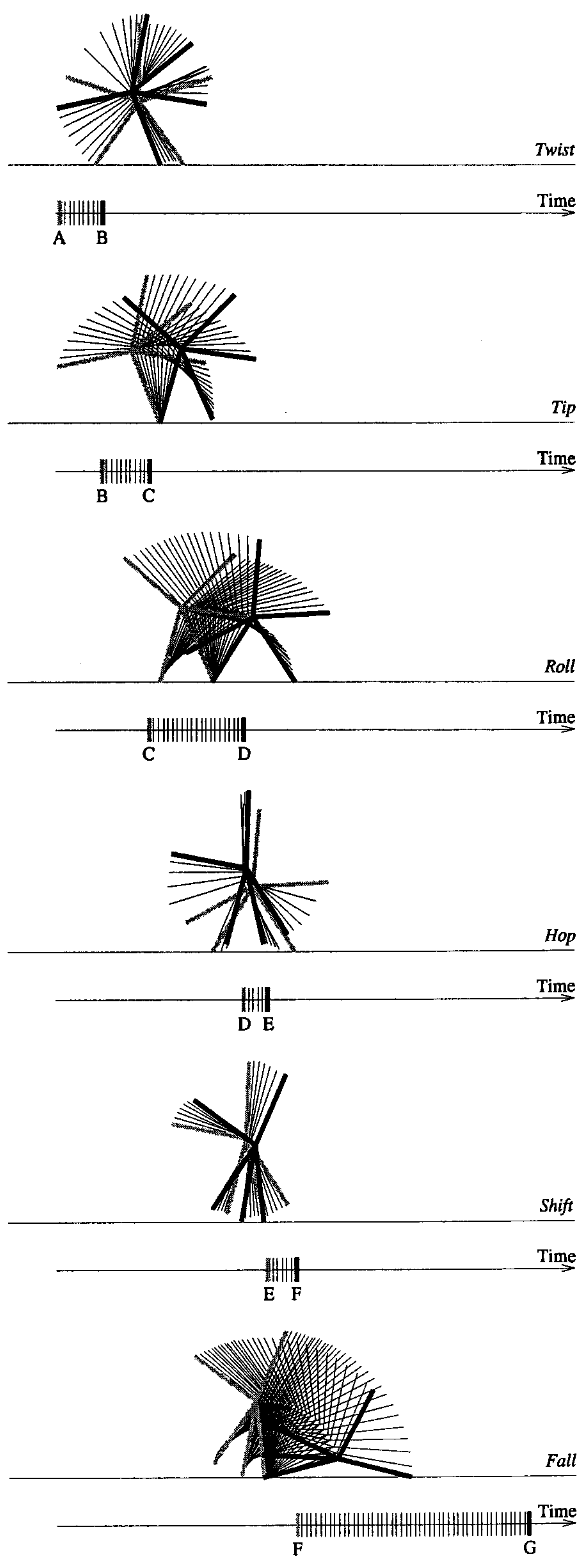


Figure 5: Mr. Star-Man begins to cartwheel, but when his time is nearly up, he falls to the finish line. The subtle posture shifts in A–B and E–F are critical to the motion.

be guaranteed to coincide with the original sense-space trajectory.

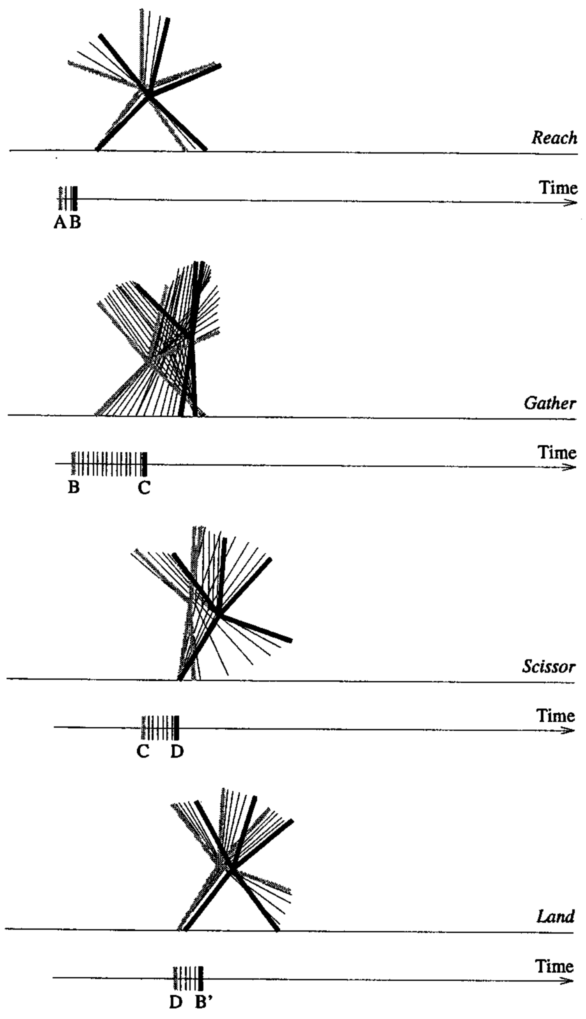


Figure 6: Mr. Star-Man shuffles. The B–C–D–B' sequence is repeated cyclically. Star-Man's stride length is limited by how long his right "foot" can stay off the ground during the step marked *Scissor*. His "arm" helps prolong this step by swinging down, offsetting the upward momentum contribution by his "leg."

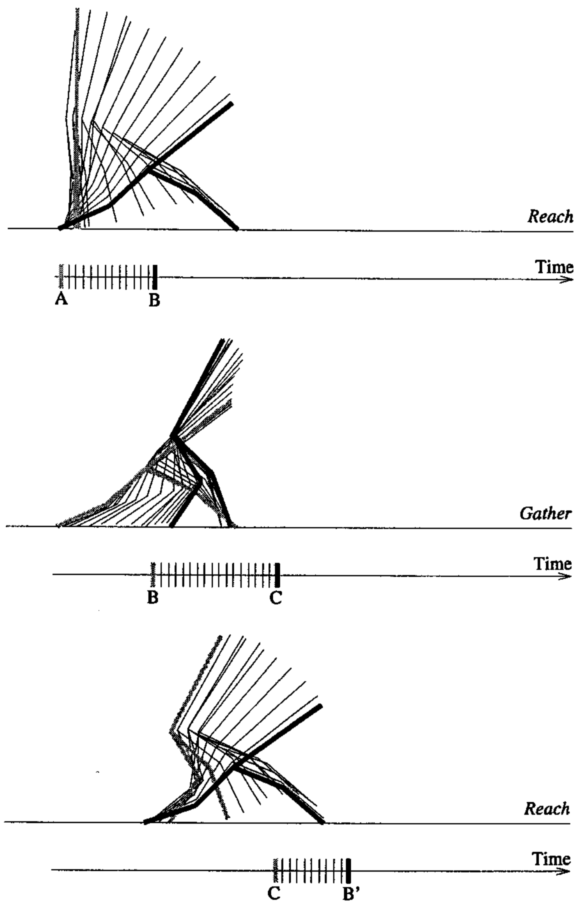


Figure 7: Beryl Biped skips. The B–C–B' sequence is repeated cyclically. Skipping differs from walking; the foot that starts out in the back never moves to the front. Balance is less of an issue in this type of skipping than in walking because intermediate postures are statically stable. However, this skipping is evidently not as fast as walking can be.

Without this restriction, freshly generated stimulus functions tend either to dominate the trajectory or not to modify it at all.

3 Results

In an earlier suite of tests involving unbranched articulated figures [13], we showed that the algorithm could cope with two-point boundary conditions as well as one-point boundary conditions with "athletic" evaluation functions (jump height and walking distance). The present test suite is a further exploration of the "walking" problem that involves bigger, more complex articulated figures, most of which are branched.

We present results for three selected runs. In the first two cases, the evaluation function was proportional to the net horizontal distance covered by the creature's center of mass in the given time. The farther the creature's center of mass moved to the right, the better the trajectory—regardless of how the movement was accomplished. In the third case, the evaluation function had to be modified slightly to obtain the desired behavior. In each case, approximately 30 to 60 minutes of elapsed time on the CM-2 was required to compute each solution shown.[4]

The first problem involves Five-Rod Fred, a creature comprised of five equal-length rods, linked consecutively. The middle rods are of equal mass, but the terminal rods are five times heavier. Each joint allows its pair of connected rods to be at most 30° from collinear. We constructed this creature in the expectation that it would behave like an inchworm. To our surprise, 64 generations of evolution produced the solution depicted in Figure 2. After 100 generations, this behavior was improved by the addition of a rolling phase at the end of the motion (Figure 3).

Fred's final behavior is generated by just five of the ten available SR pairs, which is not unusual for the behaviors described here. Ini-

[4]The quoted times are for 5 rods, 50 time steps, and 100 generations.

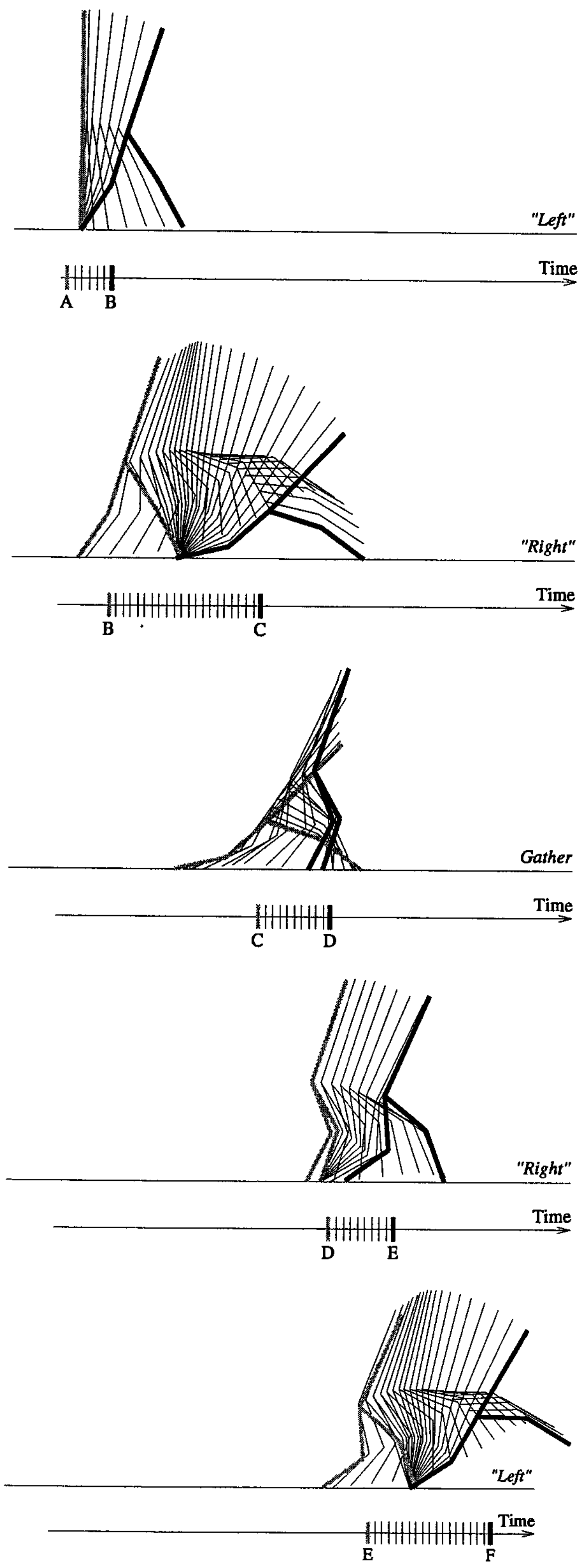


Figure 8: Beryl walks, but one step (C–D–E) is really a skip. The GA/SR combination frequently produces hybrid gaits of this type.

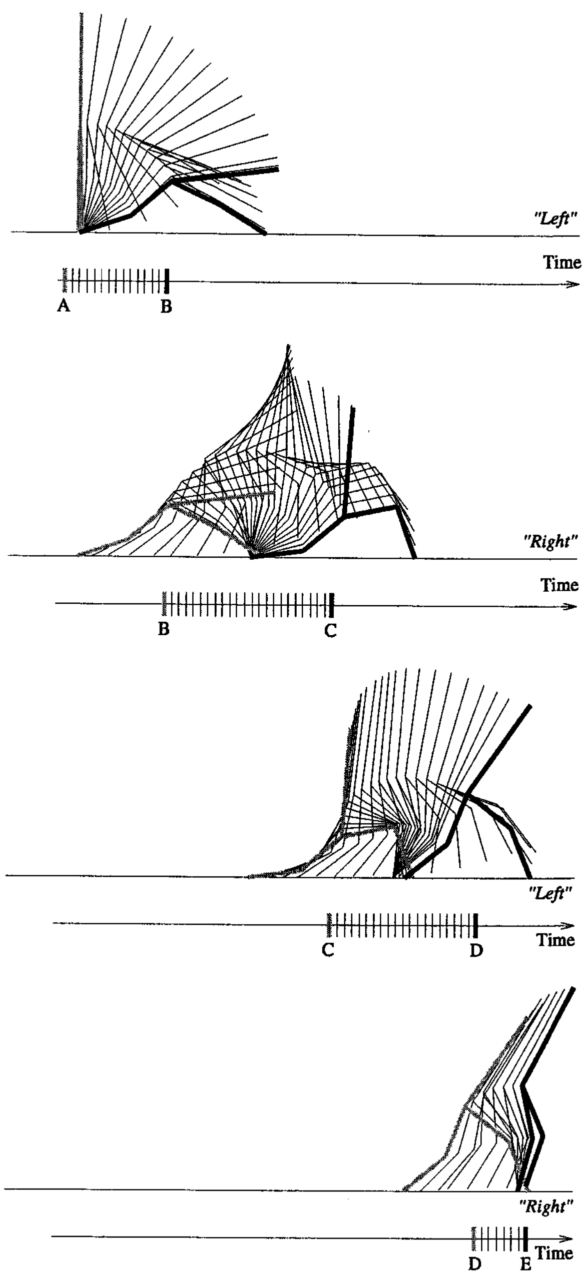


Figure 9: Beryl walks. How to modify the evaluation function to reward grace as well as distance remains an open question.

tially, two SR pairs produce the asymmetric curling motion shown in the first panel of Figure 3. Two more SR pairs produce the leaping motion. Finally, a single SR pair causes Fred to adopt an inversely curled configuration for the remainder of the trajectory.

Mr. Star-Man is another five-rod creature, but one with a different, branched topology. All of his rods are of equal length and mass. Joint-angle ranges are defined with respect to Star-Man's "torso": each "limb" rod is confined to one of the quadrants defined relative to his top "torso" rod. The trajectory in Figure 4 depicts a behavior that evolved after 20 generations: Star-Man tips over on his side, and then employs a sideways cantering motion. This motion is cyclic; the hop-land sequence (C–D–C') is repeated several times. By generation 37 a new behavior had evolved—the start of a cartwheel maneuver followed by a fall to the right (Figure 5). The best behavior after 94 generations was yet a third strategy that involved sideways shuffling (Figure 6).

Beryl Biped is a headless 2D humanoid with a rigid torso, jointed legs, point feet, and rod masses of human proportion. Despite her limitations, Beryl learns human-like locomotion. To elicit these behaviors, we had to change the evaluation function. When center-of-mass translation was used as the measure of progress, Beryl adopted the short-term strategy of plunging headlong into the ground (not shown). When the reference point was changed from the center of mass to the midpoint between her two feet, she still tried falling forward, but in 100 generations she discovered the more tactically sound gaits depicted in Figures 7–9. Thus, like any optimization procedure, the genetic algorithm will fail to find good optima if the evaluation function is sufficiently "deceptive," but simple changes to the evaluation function can often restore acceptable behavior.

4 Conclusions and future work

We have presented an effective algorithm for 2D SC problems involving articulated figures. The trajectories computed by our algorithm differ qualitatively from those that would be produced by existing local-search techniques: they are complex, varied, multi-staged, and sometimes far from obvious. Our work differs from previous global-search and learning approaches to articulated-figure motion control [6, 4, 12] in its adherence to physical law, the nature of the articulated figures being considered, and the generality of the problem statement, respectively.

Developing an algorithm of this type requires numerous decisions regarding the design of mutation and crossover operators and the assignment of parameter values. It is difficult to be sure that one has chosen optimally, since doing so is itself a combinatorial search problem! However, few of the choices are critical. In this paper and elsewhere [13] we have described what we believe are the important choices to get right.

Our current work is directed towards gaining a better understanding of how the algorithm works and making more use of its present capabilities. In particular we are investigating other stimulus-response representations, ways in which a user can influence the GA interactively, alternative search algorithms, and editing techniques whereby composite trajectories can be formed by splicing together precomputed behaviors. In the future, we hope to extend our techniques to work with 3D articulated figures.

5 Acknowledgments

We thank Tom Cheatham for use of the CM-2 at Harvard. Carter Cornwall, Dave Davis, Steven Salzberg, Stuart Shieber, and the members of the Harvard Animation Group, the MIT Leg Lab, and the Cambridge Seminar on Natural and Artificial Computation all provided useful advice and suggestions. Adam Ginsburg wrote support software for the project. JTN is grateful for a Graduate Fellowship from the Fannie and John Hertz Foundation. This work was supported in part by an NSF grant to Martin Karplus.

References

[1] D. Baraff. Coping with friction for non-penetrating rigid body simulation. *Computer Graphics*, 25(4):31–40, July 1991.

[2] L. S. Brotman and A. N. Netravali. Motion interpolation by optimal control. *Computer Graphics*, 22(4):309–315, August 1988.

[3] M. F. Cohen. Interactive spacetime control for animation. *Computer Graphics*, 26(2):293–302, July 1992.

[4] Y. Davidor. A genetic algorithm applied to robot trajectory generation. In L. Davis, editor, *Handbook of Genetic Algorithms*, chapter 12, pages 144–165. Van Nostrand Reinhold, New York, 1991.

[5] L. Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991.

[6] H. de Garis. Genetic programming: Building artificial nervous systems using genetically programmed neural network modules. In *Proceedings of the Seventh International Conference on Machine Learning*, pages 132–139, Austin, Texas, June 1990.

[7] K. A. De Jong. *An Analysis of the Behavior of a Class of Genetic Algorithms*. PhD thesis, University of Michigan, 1975.

[8] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, Massachusetts, 1988.

[9] J. K. Hahn. Realistic animation of rigid bodies. *Computer Graphics*, 22(4):299–308, August 1988.

[10] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[11] J. Luh, M. Walker, and R. Paul. On-line computational scheme for mechanical manipulators. *Trans. ASME, J. Dynamic Systems, Measurement, and Control*, 102:69–76, 1980.

[12] P. Maes and R. A. Brooks. Learning to coordinate behaviors. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 796–802, Menlo Park, California, 1990. American Association for Artificial Intelligence.

[13] J. T. Ngo and J. Marks. Physically realistic trajectory planning in animation: A stimulus-response approach. Technical Report TR-21-92, Center for Research in Computing Technology, Harvard University, October 1992.

[14] B. F. Skinner. *The Behavior of Organisms; An Experimental Analysis*. The Century Psychology Series. D. Appleton-Century, New York, London, 1938.

[15] G. Syswerda. Uniform crossover in genetic algorithms. *Proceedings of the Third International Conference on Genetic Algorithms*, pages 2–9, 1989.

[16] J. Wilhelms and R. Skinner. A "notion" for interactive behavioral animation control. *IEEE Computer Graphics and Applications*, 10(3):14–22, May 1990.

[17] A. Witkin and M. Kass. Spacetime constraints. *Computer Graphics*, 22(4):159–168, August 1988.

Animation of Plant Development

Przemyslaw Prusinkiewicz
Department of Computer Science
University of Calgary
Calgary, Alberta, Canada T2N 1N4

Mark S. Hammel
Department of Computer Science
University of Calgary
Calgary, Alberta, Canada T2N 1N4

Eric Mjolsness
Department of Computer Science
Yale University
New Haven, CT 06520-2158

ABSTRACT

This paper introduces a combined discrete/continuous model of plant development that integrates L-system-style productions and differential equations. The model is suitable for animating simulated developmental processes in a manner resembling time-lapse photography. The proposed technique is illustrated using several developmental models, including the flowering plants *Campanula rapunculoides, Lychnis coronaria*, and *Hieracium umbellatum*.

CR categories: F.4.2 [**Mathematical Logic and Formal Languages**]: Grammars and Other Rewriting Systems: *Parallel rewriting systems*, I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism: *Animation*, I.6.3 [**Simulation and Modeling**]: Applications, J.3 [**Life and Medical Sciences**]: Biology

Keywords: animation through simulation, realistic image synthesis, modeling of plants, combined discrete/continuous simulation, L-system, piecewise-continuous differential equation.

1 INTRODUCTION

Time-lapse photography reveals the enormous visual appeal of developing plants, related to the extensive changes in topology and geometry during growth. Consequently, the animation of plant development represents an attractive and challenging problem for computer graphics. Its solution may enable us to retrace the growth of organs hidden from view by protective cell layers or tissues, illustrate processes that do not produce direct visual effects, and expose aspects of development obscured in nature by concurrent phenomena, such as the extensive daily motions of leaves and flowers. Depending on the application, different degrees of realism may be sought, ranging from diagrammatic representations of developmental mechanisms to photorealistic recreations of nature's beauty.

Known techniques for simulating plant development, such as L-systems [16, 27, 28, 31], their variants proposed by Aono and Kunii [1], and the AMAP software [4, 10], operate in discrete time, which means that the state of the model is known only at fixed time intervals. This creates several problems if a smooth animation of development is sought [27, Chapter 6]:

- Although, in principle, the time interval can be arbitrarily small, once it has been chosen it becomes a part of the model and cannot be easily changed. From the viewpoint of computer animation, it is preferable to specify this interval as an easy to control parameter, decoupled from the underlying model.
- The continuity criteria responsible for the smooth progression of shapes during animation can be specified more easily in the continuous time domain.
- It is conceptually elegant to separate the model of development, defined in continuous time, from its observation, taking place in discrete intervals.

Smooth animations of plant development have been created by Miller (a growing coniferous tree [19]), Sims (artificially evolved plant-like structures [30]), and Prusinkiewicz *et. al.* (a growing herbaceous plant *Lychnis coronaria* [24]), but the underlying techniques have not been documented in the literature. Greene proposed a model of branching structures [12] suitable for animating accretive growth [11], but this model does not capture the non-accretive developmental processes observed in real plants.

This paper introduces a mathematical framework for modeling plants and simulating their development in a manner suitable for animation. The key concept is the integration of discrete and continuous aspects of model behavior into a single formalism, called *differential L-systems* (dL-systems), where L-system-style productions express qualitative changes to the model (for example, the initiation of a new branch), and differential equations capture continuous processes, such as the gradual elongation of internodes.

The proposed integration of continuous and discrete aspects of development into a single model has several predecessors.

Barzel [2] introduced *piecewise-continuous ordinary differential equations (PODEs)* as a framework for modeling processes described by differential equations with occasionally occurring discontinuities. PODEs lack a formal generative mechanism for specifying changes to system configuration resulting from discrete events, and therefore cannot be directly applied to simulate the development of organisms consisting of hundreds or thousands of modules.

Fleischer and Barr [7] addressed this limitation in a model of morphogenesis consisting of cells developing in a continuous medium.

The configuration of the system is determined implicitly by its geometry. For example, in a simulated neural network, a synapse is formed when a growing dendrite of one cell reaches another cell.

Mjolsness *et. al.* [21] pursued an alternative approach in a *connectionist model of development*. Differential equations describe the continuous aspects of cell behavior during interphase (time between cell divisions), while productions inspired by L-systems specify changes to the system configuration resulting from cell division and death. The connectionist model makes it possible to consider networks with arbitrary topology (not limited to branching structures), but requires productions that operate globally on the entire set of cells constituting the model. This puts a practical limit on the number of components in the system.

Fracchia *et. al.* [9] (see also [27, Chapter 7]) animated the development of cellular layers using a physically-based model in which differential equations simulate cell growth during the interphase, and productions of a *map L-system* capture cell divisions. The productions operate locally on individual cells, making it possible to simulate the development of arbitrarily large layers using a finite number of rules. Unfortunately, this technique does not seem to extend beyond the modeling of cellular layers.

Timed L-systems [27, Chapter 6] were introduced specifically as a formal framework for constructing models of branching structures developing in continuous time. They operate under the assumption that no information exchange between coexisting modules takes place. This is a severe limitation, as interactions between the modules are known to play an important role in the development of many plant species [14, 27, 28]. A practical application of timed L-systems to animation is described by Noser *et. al.* [22].

The model of development proposed in this paper combines elements of PODEs, the connectionist model, and L-systems. The necessary background in L-systems is presented in Section 2. Sections 3 and 4 introduce the definition of differential L-systems and illustrate it using two simple examples. Section 5 applies combined discrete/continuous simulation techniques to evaluate dL-systems over time. Section 6 focuses on growth functions, which characterize continuous aspects of model development. Application of differential L-systems to the animation of the development of higher plants is presented in Section 7, using the models of a compound leaf and three herbaceous plants as examples. A summary of the results and a list of open problems conclude the paper.

2 L-SYSTEMS

An extensive exposition of L-systems applied to the modeling of plants is given in [27]. Below we summarize the main features of L-systems pertinent to the present paper.

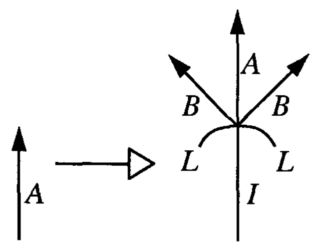


Figure 1: Example of a typical L-system production

We view a plant as a linear or branching structure composed of repeated units called *modules*. An L-system describes the development of this structure in terms of *rewriting rules* or *productions*, each of which replaces the *predecessor* module by zero, one, or more *successor* modules. For example, the production in Figure 1 replaces apex A by a structure consisting of a new apex A, an internode I, and two lateral apices B supported by leaves L.

In general, productions can be *context free* and depend only on the replaced module, or *context-sensitive* and depend also on the neighborhood of this module. A *developmental sequence* is generated by repeatedly applying productions to the consecutively obtained structures. In each step, productions are applied in parallel to all parts of the structure obtained so far.

The original formalism of L-systems [16] has a threefold discrete character [17]: the modeled structure is a finite collection of modules, each of these modules is in one of a finite number of states, and the development is simulated in discrete derivation steps. An extension called *parametric L-systems* [25, 27] increases the expressive power of L-systems by introducing a continuous characterization of the module states. Each module is represented by an identifier denoting the module *type* (one or more symbols starting with a letter) and a *state vector* of zero, one, or more numerical *parameters*. For instance, $M = A(5, 9.5)$ denotes a module M of type A with two parameters $w_1 = 5$ and $w_2 = 9.5$, forming the vector $\mathbf{w} = (5, 9.5)$. The interpretation of parameters depends on the semantics of the module definition, and may vary from one module type to another. For example, parameters may quantify the shape of the module, its age, and the concentration of substances contained within it.

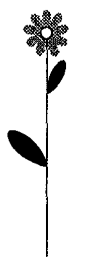


Figure 2: Turtle interpretation of a sample string

In the formalism of L-systems, modeled structures are represented as *strings* of modules. Branching structures are captured using *bracketed* strings, with the matching pairs of brackets [and] delimiting branches. We visualize these structures using a *turtle interpretation* of strings [23, 28], extended to strings of modules with parameters in [13, 25, 27]. A predefined interpretation is assigned to a set of reserved modules. Some of them represent physical parts of the modeled plant, for example a leaf or an internode, while others represent local properties, such as the magnitude of a branching angle. Reserved modules frequently used in this paper are listed below:

$F(x)$	line segment of length x,
$+(\alpha), -(\alpha)$	orientation change of the following line by $\pm\alpha$ degrees with respect to the preceding line,
$@X(s)$	a predefined surface X scaled by the factor s.

The interpretation of a string of modules proceeds by scanning it from left to right and considering the reserved modules as commands that maneuver a LOGO-style turtle. For example, Figure 2 shows the turtle interpretation of a sample string:

$$F(1)[+(45)@L(0.75)]F(0.8)[-(30)@L(0.5)]F(0.6)@K(1),$$

where symbols $@L$ and $@K$ denote predefined surfaces depicting a leaf and a flower.

3 DEFINITION OF dL-SYSTEMS

Differential L-systems extend parametric L-systems by introducing continuous time flow in place of a sequence of discrete derivation steps. As long as the parameters $\mathbf{w}$ of a module $A(\mathbf{w})$ remain in the *domain of legal values* $\mathcal{D}_A$, the module develops in a continuous way. Once the parameter values reach the boundary C_A of the domain $\mathcal{D}_A$, a production replaces module $A(\mathbf{w})$ by its descendants in a discrete event. The form of this production may depend on which segment C_{A_k} of the boundary of $\mathcal{D}_A$ has been crossed.

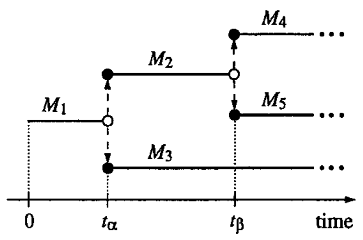


Figure 3: Fragment of the lineage tree of a hypothetical modular structure

For example, module M_2 in Figure 3 is created at time t_α as one of two descendants of the initial module M_1. It develops in the interval $[t_\alpha, t_\beta)$, and ceases to exist at time t_β, giving rise to two new modules M_4 and M_5. The instant t_β is the time at which parameters of M_2 reach the boundary of its domain of legal states $\mathcal{D}$. A hypothetical trajectory of module M_2 in its parameter space is depicted in Figure 4.

In order to formalize the above description, let us assume that the modeled structure consists of a sequence of modules (an extension to branching structures is straightforward if a proper definition of context is used [27, 28]). The state of the structure at time t is represented as a string:

$$\mu = A_1(\mathbf{w}_1)A_2(\mathbf{w}_2)\cdots A_n(\mathbf{w}_n).$$

The module $A_{i-1}(\mathbf{w}_{i-1})$ immediately preceding a given module $A_i(\mathbf{w}_i)$ in the string μ is called the *left neighbor* or *left context* of $A_i(\mathbf{w}_i)$, and the module $A_{i+1}(\mathbf{w}_{i+1})$ immediately following $A_i(\mathbf{w}_i)$ is called its *right neighbor* or *right context.* When it is inconvenient to list the indices, we use the symbols $<$, $>$, and/or subscripts l, r to specify the context of $A(\mathbf{w})$, as in the expression:

$$A_l(\mathbf{w}_l) < A(\mathbf{w}) > A_r(\mathbf{w}_r).$$

The continuous behavior of $A(\mathbf{w})$ is described by an *ordinary differential equation* that determines the rate of change $d\mathbf{w}/dt$ of parameters $\mathbf{w}$ as a function of the current value of these parameters and those of the module's neighbors:

$$\frac{d\mathbf{w}}{dt} = f_A(\mathbf{w}_l, \mathbf{w}, \mathbf{w}_r).$$

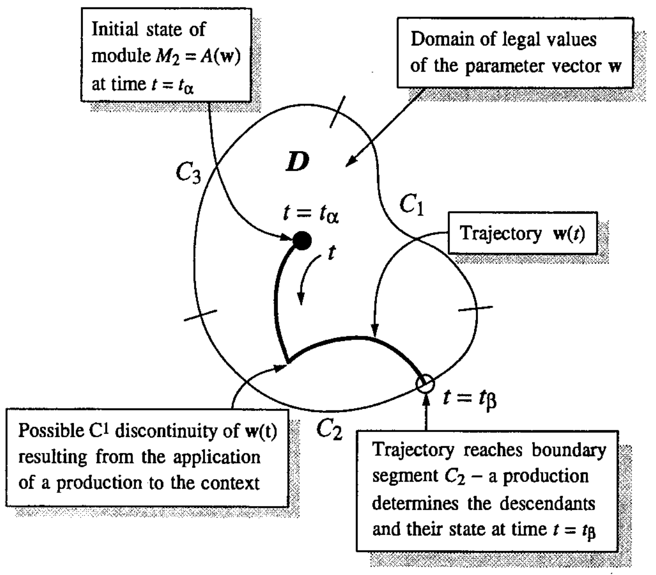
The above equation applies as long as the parameters $\mathbf{w}$ are in the domain $\mathcal{D}_A$ characteristic to the module type A. We assume that $\mathcal{D}_A$ is an open set, and specify its boundary C_A as the union of a finite number $m \geq 1$ of nonintersecting segments C_{A_k}, $k = 1, 2, \ldots, m$. The time t_β at which the trajectory of module $A(\mathbf{w})$ reaches a segment C_{A_k} of the boundary of $\mathcal{D}_A$ satisfies the expression:

$$\lim_{t \to t_\beta^-} \mathbf{w}(t) \in C_{A_k}.$$

The replacement of module $A(\mathbf{w})$ by its descendants at time t_β is described by a *production*:

$$p_{A_k} : A_l(\mathbf{w}_l) < A(\mathbf{w}) > A_r(\mathbf{w}_r) \longrightarrow$$
$$B_{k,1}(\mathbf{w}_{k,1})B_{k,2}(\mathbf{w}_{k,2})\cdots B_{k,m_k}(\mathbf{w}_{k,m_k}).$$

The module $A(\mathbf{w})$ is called the *strict predecessor* and the sequence of modules $B_{k,1}(\mathbf{w}_{k,1})B_{k,2}(\mathbf{w}_{k,2})\cdots B_{k,m_k}(\mathbf{w}_{k,m_k})$ is called the *successor* of this production. The index k emphasizes that different productions can be associated with individual segments C_{A_k} of the boundary C_A. The initial value of parameters assigned to a module $B_{k,j}(\mathbf{w}_{k,j})$ upon its creation is determined by a function $h_{A_{k,j}}$ which takes as its arguments the values of the parameters $\mathbf{w}_l$, $\mathbf{w}$, and $\mathbf{w}_r$ at the time immediately preceding production application:



Initial state of module $M_2 = A(\mathbf{w})$ at time $t = t_\alpha$
Domain of legal values of the parameter vector $\mathbf{w}$
C_3
D
$t = t_\alpha$
C_1
Trajectory $\mathbf{w}(t)$
t
$t = t_\beta$
Possible C¹ discontinuity of $\mathbf{w}(t)$ resulting from the application of a production to the context
C_2
Trajectory reaches boundary segment C_2 – a production determines the descendants and their state at time $t = t_\beta$

Figure 4: A hypothetical trajectory of module M_2 in its parameter space

$$\mathbf{w}_{k,j} = \lim_{t \to t_\beta^-} h_{A_{k,j}}(\mathbf{w}_l(t), \mathbf{w}(t), \mathbf{w}_r(t)).$$

The vector $\mathbf{w}_{k,j}$ must belong to the domain $\mathcal{D}_{B_{k,j}}$. (A stronger condition is needed to insure that the number of productions applied in any finite interval $[t, t + \Delta t]$ will be finite.)

In summary, a differential L-system is defined by the initial string of modules μ_0 and the specification of each module type under consideration. The specification of a module type A consists of four components:

$$< \mathcal{D}_A, C_A, f_A, P_A >,$$

where:

- the open set $\mathcal{D}_A$ is the domain of legal parameter values of modules of type A,
- the set $C_A = C_{A_1} \cup \ldots \cup C_{A_m}$ is the boundary of $\mathcal{D}_A$, consisting of nonintersecting segments $C_{A_1}, \ldots, C_{A_m}$,
- the function f_A specifies a system of differential equations that describe the continuous behavior of modules of type A in their domain of legal parameter values $\mathcal{D}_A$,
- the set of productions $P_A = \{p_{A_1}, \ldots, p_{A_m}\}$ captures the discrete behavior of modules of type A.

A production $p_{A_k} \in P_A$ is applied when the parameters of a module M of type A reach segment C_{A_k} of the boundary C_A. At this time module M disappears, and zero, one, or more descendant modules are created. The functions $h_{A_{k,j}}$ embedded in productions p_{A_k} determine the initial values of parameters in the successor modules.

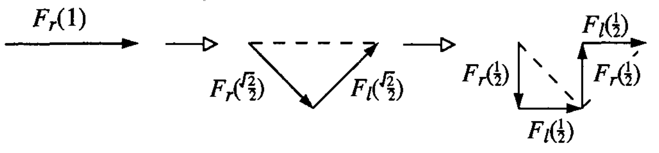


Figure 5: Initial steps in the construction of a dragon curve

4 EXAMPLES OF dL-SYSTEMS

We will illustrate the notion of a dL-system using two sample models suitable for animating the development of the *dragon curve* and the filamentous alga *Anabaena catenula.*

4.1 A dL-system model of the dragon curve

In the discrete case, consecutive iterations of the *dragon curve* (described, for example, in [27, Chapter 1]) can be obtained by the following parametric L-system:

$$\begin{aligned} \omega &: \; --F_r(1) \\ p_1 &: \; F_r(s) \rightarrow -F_r(s\tfrac{\sqrt{2}}{2}) + +F_l(s\tfrac{\sqrt{2}}{2})- \\ p_2 &: \; F_l(s) \rightarrow +F_r(s\tfrac{\sqrt{2}}{2}) - -F_l(s\tfrac{\sqrt{2}}{2})+ \end{aligned}$$

Assuming that symbols $+$ and $-$ represent turns of $\pm 45°$, this L-system encodes a Koch construction [18, Chapter 6] that repeatedly substitutes sides of an isosceles right-angled triangle for its hypotenuse (Figure 5). Subscripts l and r indicate that the triangle is formed respectively on the left or right side of the oriented predecessor segment. A corresponding dL-system that generates the dragon curve through the continuous progression of shapes indicated in Figure 6 is given below:

```
initial string: $--F_r(1,1)$
$F_r(x,s)$ :
    if $x < s$ solve $\frac{dx}{dt} = \frac{s}{T}, \frac{ds}{dt} = 0$
    if $x = s$ produce $-F_r(0, s\frac{\sqrt{2}}{2}) + F_h(s,s) + F_l(0, s\frac{\sqrt{2}}{2})-$
$F_l(x,s)$ :
    if $x < s$ solve $\frac{dx}{dt} = \frac{s}{T}, \frac{ds}{dt} = 0$
    if $x = s$ produce $+F_r(0, s\frac{\sqrt{2}}{2}) - F_h(s,s) - F_l(0, s\frac{\sqrt{2}}{2})+$
$F_h(x,s)$ :
    if $x > 0$ solve $\frac{dx}{dt} = -\frac{s}{T}, \frac{ds}{dt} = 0$
    if $x = 0$ produce $\varepsilon$
```

The operation of this model starts with the replacement of the initial module $F_r(1,1)$ with the string:

$$-F_r(0, \frac{\sqrt{2}}{2}) + F_h(1,1) + F_l(0, \frac{\sqrt{2}}{2})-,$$

which has the same turtle interpretation: a line segment of unit length [1]. Next, the horizontal line segment represented by module F_h decreases in length with the speed $\frac{dx}{dt} = -\frac{1}{T}$, while the diagonal segments represented by modules F_r and F_l elongate with the speed $\frac{dx}{dt} = \frac{\sqrt{2}}{2}\frac{1}{T}$. The constant T determines the lifetime of the modules: after time T, the module F_h reaches zero length and is removed from the string (replaced by the empty string ε), while both modules F_r and F_l reach their maximum length of $\frac{\sqrt{2}}{2}$ and are replaced by their respective successors. These successors subsequently follow the same developmental pattern.

It is not accidental that the predecessor and the successor of the productions for $F_r(x,s)$ and $F_l(x,s)$ have identical geometric interpretations. Since productions are assumed to be applied instantaneously, any change of the model's geometry introduced by a production would appear as a discontinuity in the animation. In general, correctly specified productions satisfy *continuity criteria* [27, Chapter 6], which means that they conserve physical entities such as shape, mass, and velocity of modules.

4.2 A dL-system model of *Anabaena catenula*

The continuously developing dragon curve has been captured by a context-free dL-system, in which all productions and equations depend only on the strict predecessor module. A simple example of a context-sensitive model inspired by the development of the blue-green alga *Anabaena catenula* [3, 20, 27] is given below.

Anabaena forms a nonbranching filament consisting of two classes of cells: *vegetative cells* and *heterocysts*. A vegetative cell usually divides into two descendant vegetative cells. However, in some cases a vegetative cell differentiates into a heterocyst. The spacing between heterocysts is relatively constant, in spite of the continuing growth of the filament. Mathematical models explain this phenomenon using a biologically motivated hypothesis that the distribution of heterocysts is regulated by nitrogen compounds produced by the heterocysts, diffusing from cell to cell along the filament, and decaying in the vegetative cells. If the compound concentration in a vegetative cell falls below a specific level, this cell differentiates into a heterocyst (additional factors are captured by more sophisticated models). A model operating in continuous time according to this description can be captured by the following dL-system:

```
initial string: $F_h(x_{max}, c_{max})F_v(x_{max}, c_{max})F_h(x_{max}, c_{max})$
$F(x_l, c_l) < F_v(x,c) > F(x_r, c_r)$ :
    if $x < x_{max}$ & $c > c_{min}$
        solve $\frac{dx}{dt} = rx, \frac{dc}{dt} = D \cdot (c_l + c_r - 2c) - \mu c$
    if $x = x_{max}$ & $c > c_{min}$
        produce $F_v(kx_{max}, c)F_v((1-k)x_{max}, c)$
    if $c = c_{min}$
        produce $F_h(x,c)$
$F_h(x,c)$:
        solve $\frac{dx}{dt} = r_x(x_{max} - x), \frac{dc}{dt} = r_c(c_{max} - c)$
```

Vegetative cells F_v and heterocysts F_h are characterized by their length x and concentration of nitrogen compounds c. The differential equations for the vegetative cell F_v indicate that while the cell length x is below the maximum value x_{max} and the compound concentration c is above the threshold c_{min}, the cell elongates exponentially according to the equation $\frac{dx}{dt} = rx$, and the compound concentration changes according to the equation:

$$\frac{dc}{dt} = D \cdot (c_l + c_r - 2c) - \mu c.$$

The first term in this equation describes diffusion of the compounds through the cell walls. Following *Fick's law* [5, page 404], the

[1]The turtle interprets the first parameter as the segment length, and ignores the second parameter.

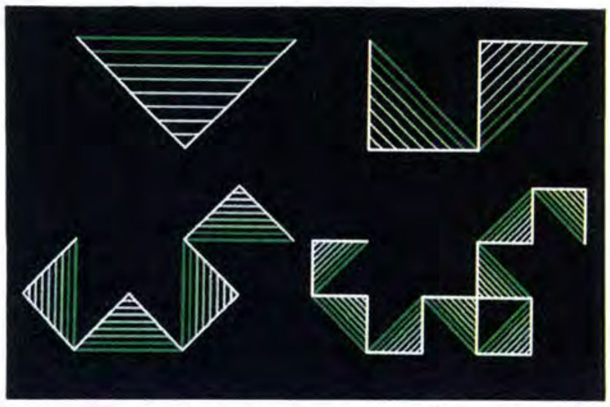


Figure 6: Development of the dragon curve simulated using a dL-system, recorded in time intervals $\Delta t = \frac{1}{8}T$. Top left: Superimposed stages $0 - 8$, top right: stages $8 - 16$, bottom row: stages $16 - 24$ and $24 - 32$.

rate of diffusion is proportional to the differences of compound concentrations, $c_r - c$ and $c_l - c$, between the neighbor cells and the cell under consideration. The term μc describes exponential decay of the compounds in the cell.

In addition to the differential equations, two productions describe the behavior of a vegetative cell. If the cell reaches maximum length x_{max} while the concentration c is still above the threshold c_{min}, the cell divides into two vegetative cells of length kx_{max} and $(1-k)x_{max}$, with the compound concentration c inherited from their parent cell. Otherwise, if the concentration c drops down to the threshold c_{min}, the cell differentiates into a heterocyst. Both productions satisfy the continuity criteria by conserving total cell length and concentration of nitrogen compounds.

The last line of the model specifies the behavior of the heterocysts. Their length and compound concentration converge exponentially to the limit values of x_{max} and c_{max}. The heterocysts do not undergo any further transformations.

Simulation results obtained using the above model are shown in Figure 7. The cells in the filament are represented as horizontal line segments with the colors indicating the concentration of nitrogen compounds. Consecutive developmental stages are drawn one under another. An approximately equal spacing between the heterocysts (shown in white) is maintained for any horizontal section, as postulated during model formulation.

Note that for incorrectly chosen constants in the model, the spacing between heterocysts may be distorted; for example, groups of adjacent vegetative cells may almost simultaneously differentiate into heterocysts.

5 EVALUATION OF dL-SYSTEMS

Although Figures 6 and 7 were obtained using dL-systems, we have not yet discussed the techniques needed to *evaluate* them. This term denotes the calculation of the sequence of strings $\mu(0) = \mu_0$, $\mu(\Delta t) = \mu_1$, $\ldots$, $\mu(n\Delta t) = \mu_n$ representing the states of the modeled structure at the desired intervals Δt. We address the problem of dL-system evaluation in the framework of the combined discrete/continuous paradigm for system simulation introduced by Fahrland [6] and presented in a tutorial manner by Kreutzer [15].

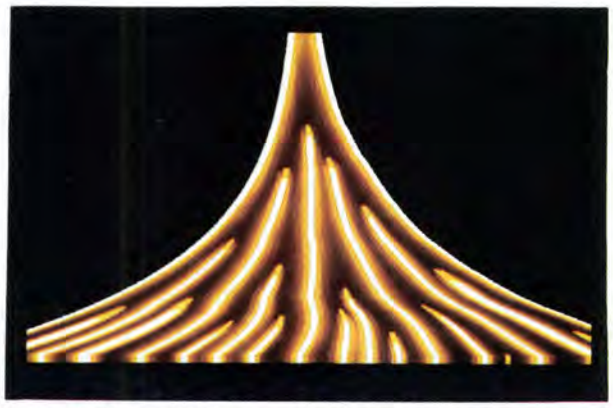


Figure 7: Diagrammatic representation of the development of *Anabaena catenula*, simulated using a dL-system with the constants set to the following values: $x_{max} = 1$, $c_{max} = 255$, $c_{min} = 5$, $D = \mu = 0.03$, $r = 1.01$, $k = 0.37$, $r_x = 0.1$, $r_c = 0.15$. The development was recorded from $t_{min} = 200$ to $t_{max} = 575$ at the intervals $\Delta t = 1$. Developmental stages are shown as horizontal lines with the colors indicating the concentration c of nitrogen compounds. Dark brown represents c_{min}; white represents c_{max}.

According to this paradigm, the evaluation can be viewed as a *dynamic process* governed by a *scheduler*: a part of the simulation program that monitors the state of the model, advances time, and dispatches the activities to be performed. In the absence of discrete events (productions), the scheduler repeatedly advances time by the *time slice* Δt. During each slice, the differential equations associated with the modules are integrated numerically (using an integration technique appropriate for the equations in hand), thus advancing the state of the structure from $\mu(t)$ to $\mu(t + \Delta t)$. If the scheduler detects that a discrete event should occur (i.e., a production should be applied) at time t' within the interval $[t, t + \Delta t)$, this interval is divided into two subintervals $[t, t')$ and $[t', t + \Delta t)$. The differential equations are integrated in the interval $[t, t')$ and yield parameter values for the production application at time t'. The production determines the initial values for the differential equations associated with the newly created modules; these equations are integrated in the remaining interval $[t', t + \Delta t)$. Each of the intervals $[t, t')$ and $[t', t + \Delta t)$ is subdivided further if more discrete events occur during $[t, t + \Delta t)$.

Plant structures generated using dL-systems may consist of large numbers (thousands) of modules. If many modules are replaced at different times t' during the interval $[t, t + \Delta t)$, the global advancement of time may require an excessive subdivision of this interval, leading to a slow evaluation of the model. This problem can be solved by detecting and processing events the interval $[t, t + \Delta t)$ individually for each module. The increase of simulation speed is obtained at the expense of accuracy, since the state of the context of a module replaced at time $t' \in (t, t + \Delta t)$ must be approximated, for example, by its state at time t. No accuracy is lost in the context-free case.

In the above description we assumed that the scheduler is capable of detecting each instant t' at which a discrete event occurs. If the differential equations are sufficiently simple, we can solve them analytically and determine time t' explicitly. In general, we need numerical techniques for special event location in piecewise-continuous ordinary differential equations, as described by Shampine *et. al.* [29], and Barzel [2, Appendix C].

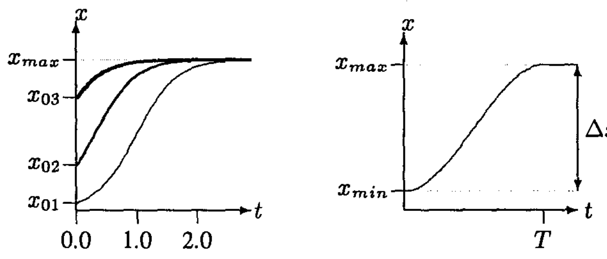


Figure 8: Examples of sigmoidal growth functions. a) A family of logistic functions plotted using $r = 3.0$ for different initial values x_0. b) A cubic function $G_{\Delta x, T}$.

6 GROWTH FUNCTIONS

Growth functions describe continuous processes such as the expansion of individual cells, elongation of internodes, and gradual increase of branching angles over time. For example, the differential equations included in the dL-system for the dragon curve (Section 4.1) describe linear elongation of segments F_r and F_l, and linear decrease in length of segments F_h. The dL-system model of *Anabaena* (Section 4.2) assumes exponential elongation of cells.

In higher plants, the growth functions are often of *sigmoidal* (S-shaped) type, which means that they initially increase in value slowly, then accelerate, and eventually level off at or near the maximum value. A popular example of a sigmoidal function is Velhurst's *logistic* function (c.f. [5, page 212]), defined by the equation:

$$\frac{dx}{dt} = r\left(1 - \frac{x}{x_{max}}\right)x$$

with a properly chosen initial value x_0 (Figure 8a). Specifically, x_0 must be greater than zero, which means that neither the initial length nor the initial growth rate of a module described by the logistic function will be equal to zero. In order to obtain a continuous progression of forms, it is often convenient to use a growth function that has zero growth rates at both ends of an interval T within which its value increases from x_{min} (possibly zero) to x_{max}. These requirements can be satisfied, for example, by a *cubic* function of time. Using the Hermite form of curve specification [8, page 484], we obtain:

$$x(t) = -2\frac{\Delta x}{T^3}t^3 + 3\frac{\Delta x}{T^2}t^2 + x_{min},$$

where $\Delta x = x_{max} - x_{min}$ and $t \in [0, T]$. The equivalent differential equation is:

$$\frac{dx}{dt} = -6\frac{\Delta x}{T^3}t^2 + 6\frac{\Delta x}{T^2}t = 6\frac{\Delta x}{T^2}\left(1 - \frac{t}{T}\right)t$$

with the initial condition $x_0 = x_{min}$. In order to extend this curve to infinity (Figure 8b), we define:

$$\frac{dx}{dt} = G_{\Delta x, T}(t) = \begin{cases} 6\frac{\Delta x}{T^2}\left(1 - \frac{t}{T}\right)t & \text{for } t \in [0, T] \\ 0 & \text{for } t \in (T, +\infty). \end{cases}$$

Although the explicit dependence of the function G on time is questionable from the biological point of view (a plant module does not have a means for measuring time directly), parametric cubic functions constitute a well understood computer graphics tool [8, Chapter 11.2] and can be conveniently used to approximate the observed changes of parameter values over time.

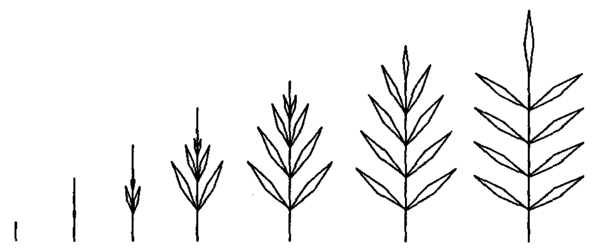


Figure 9: Development of a compound leaf simulated using a dL-system. Parameter values are: $n_0 = 4$, $x_0 = 1.0$, $x_{th} = 2.0$, $k = 0.5$, $r_a = 2.0$, $x_{amax} = 3.0$, $r_i = 1.0$, $x_{imax} = 3.0$, $s_0 = 0.05$, $r_s = 2.0$, $s_{max} = 6.0$, $\alpha_0 = 2.0$, $r_\alpha = 1.0$, $\alpha_{max} = 60.0$, and $\Delta t = 0.01$. The stages shown represent frames 50, 215, 300, 400, 500, 600, and 900 of an animated sequence.

7 MODELING OF HIGHER PLANTS

In this section we present sample applications of dL-systems to the animation of the development of higher plants.

7.1 Pinnate Leaf

A pinnate leaf provides a simple example of a *monopodial* branching structure. *Monopodial* branching occurs when the apex of the main axis produces a succession of *nodes* bearing *organs* — leaves or flowers — which are separated by *internodes*. In the case of pinnate leaves with the leaflets occurring in pairs (termed *opposite* arrangement), the essence of this process can be captured by the L-system production [27, page 71]:

$$F_a \longrightarrow F_i[+@L][-@L]F_a,$$

where F_a denotes the apex, F_i — an internode, and $@L$ — a leaflet. The dL-system model given below extends this L-system with growth functions that control the expansion of all components and gradually increase branching angles over time.

```
initial string: $F_a(x_0, n_0)$
$F_a(x, n)$ :
    if $x < x_{th}$
        solve $\frac{dx}{dt} = r_a\left(1 - \frac{x}{x_{amax}}\right)x,\ \frac{dn}{dt} = 0$
    if $x = x_{th}$ & $n > 0$
        produce $F_i(kx)[+(\alpha_0)@L(s_0)][-(\alpha_0)@L(s_0)]$
                $F_a((1-k)x, n-1)$
    if $x = x_{th}$ & $n = 0$
        produce $F_i(x)@L(s_0)$
$F_i(x)$ :   solve $\frac{dx}{dt} = r_i\left(1 - \frac{x}{x_{imax}}\right)x$
$L(s)$ :   solve $\frac{ds}{dt} = r_s\left(1 - \frac{s}{s_{max}}\right)s$
$\pm(\alpha)$ :   solve $\frac{d\alpha}{dt} = r_\alpha\left(1 - \frac{\alpha}{\alpha_{max}}\right)\alpha$
```

The apex F_a has two parameters x and n which indicate its current length and the remaining number of internodes to be produced. The apex elongates according to the logistic function with parameters r (controlling growth rate) and x_{amax} (controlling the asymptotic apex length). Upon reaching the threshold length x_{th}, the apex produces a pair of leaflets $@L$ and subdivides into an internode F_i of length kx and a shorter apex of length $(1-k)x$. Once the predefined number n_0 of leaf pairs have been created, the apex

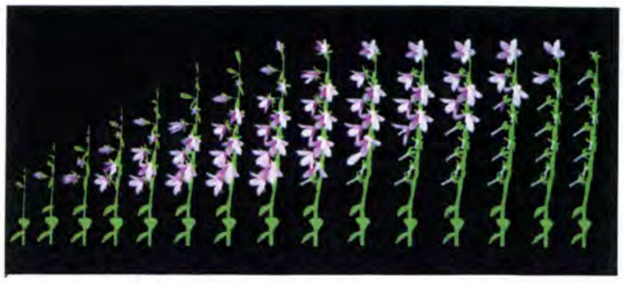


Figure 10: Development of the herbaceous plant *Campanula rapunculoides*. The snapshots show every 25th frame of a computer animation, starting with frame 175.

transforms itself into an internode and produces the terminal leaflet. The length of internodes, the size of leaflets, and the magnitude of the branching angles increase according to the logistic functions. Snapshots of the leaf development simulated by the above model are shown in Figure 9.

7.2 *Campanula rapunculoides*

The inflorescence of *Campanula rapunculoides* (creeping bellflower) has a monopodial branching structure similar to that of a pinnate leaf; consequently, it is modeled by a similar dL-system:

```
initial string: $F_a(x_0, n_0)$
$F_a(x, n)$ :
    if $x < x_{th}$
        solve $\frac{dx}{dt} = v$, $\frac{dn}{dt} = 0$
    if $x = x_{th}$ & $n > 0$
        produce $F_i(kx)[+(\alpha_0)@K]F_a((1-k)x, n-1)$
    if $x = x_{th}$ & $n = 0$
        produce $F_i(x)@K$
$F_i(x)$ :   solve $\frac{dx}{dt} = G_{\Delta x, T_1}(t)$
$+(\alpha)$ : solve $\frac{d\alpha}{dt} = G_{\Delta\alpha, T_2}(t)$
```

The apex is assumed to grow at a constant speed. Cubic growth functions describe the elongation of internodes and the gradual increase of branching angles. The combination of the linear growth of the apex with the cubic growth of the internodes results in first-order continuity of the entire plant height (except when apex F_a is transformed into internode F_i and terminal flower $@K$).

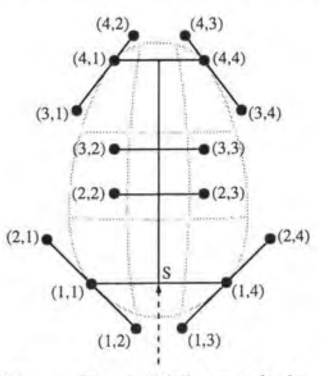


Figure 11: A Bézier patch defined by a branching structure

Figure 10 presents a sequence of snapshots from an animation of *Campanula*'s development. It was obtained using the above dL-system augmented with rules that govern the development of flowers $@K$ from a bud to an open flower to a fruit. The petals and sepals have been modeled as Bézier patches, specified by control points placed at the ends of simple branching structures (Figure 11). Each structure is attached to the remainder of the model at point S. The lengths of the line segments and the magnitudes of the branching angles have been controlled by cubic growth functions, yielding the developmental sequence shown in Figure 12. When the flower transforms into a fruit, productions instantaneously remove the petals from the model (it is assumed that the time over which a petal falls off is negligible compared to the time slice used for the animation of development). Manipulation of Bézier patches using L-systems has been described in detail by Hanan [13].

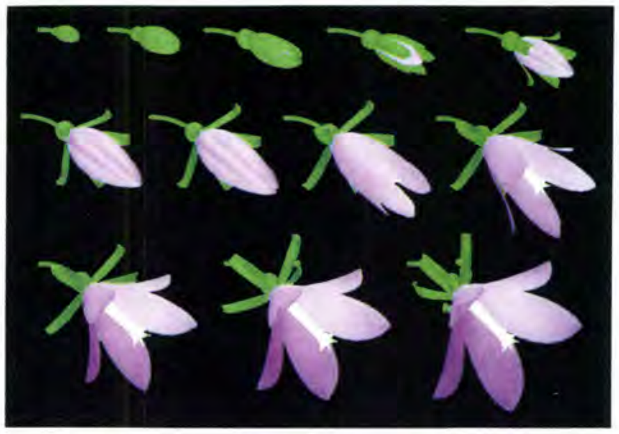


Figure 12: Development of a single flower of *Campanula rapunculoides*

7.3 *Lychnis coronaria*

The inflorescence of *Lychnis coronaria* (rose campion) is an example of a sympodial branching structure, characterized by large branches that carry the main thrust of development. As presented in [27, page 82] and [28], the apex of the main axis turns into a flower shortly after the initiation of a pair of lateral branches. Their apices turn into flowers as well, and second-order branches take over. The lateral branches originating at a common node develop at the same rate, but the development of one side is delayed with respect to the other. This process repeats recursively, as indicated by the following L-system:

$$\begin{array}{lll} \omega : & A_7 & \\ p_1 : & A_7 \longrightarrow F[A_0][A_4]F@K & \\ p_2 : & A_i \longrightarrow A_{i+1} & 0 \le i < 7 \end{array}$$

Production p_1 shows that, at their creation time, the lateral apices have different states A_0 and A_4. Consequently, the first apex requires eight derivation steps to produce flower $@K$ and initiate a new pair of branches, while the second requires only four steps.

A corresponding dL-system using cubic growth functions to describe the elongation of internodes F is given below:

```
initial string: $A(\tau_{max})$
$A(\tau)$ :
    if $\tau < \tau_{max}$ solve $\frac{d\tau}{dt} = 1$
    if $\tau = \tau_{max}$ produce $F(0)[A(0)][A(\frac{\tau_{max}}{2})]F(0)@K$
$F(x)$ :             solve $\frac{dx}{dt} = G_{\Delta x, T}(t)$
```

For simplicity, we have omitted leaves and symbols controlling the relative orientation of branches in space. The operation of the model

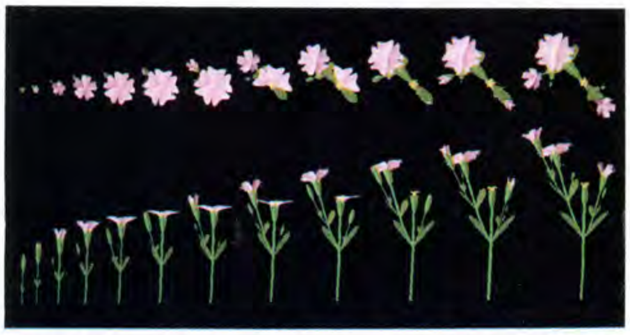


Figure 13: Development of *Lychnis coronaria*. The snapshots show every 25th frame of a computer animation, starting with frame 150.

is governed by apices A characterized by their age τ and assumed to have negligible size. Upon reaching the maximum age τ_{max}, an apex splits into two internodes F, creates two lateral apices A with different initial age values 0 and $\frac{\tau_{max}}{2}$, and initiates flower $@K$. In order to satisfy continuity criteria, the initial length of internodes is assumed to be zero.

Figure 13 shows selected snapshots from an animation of the development of *Lychnis* obtained using an extension of this dL-system. As in *Campanula*, the individual flowers have been modeled using Bézier patches controlled by the dL-system.

7.4 *Hieracium umbellatum*

The compound leaf and the inflorescences of *Campanula* and *Lychnis* have been captured by context-free dL-systems, assuming no flow of information between coexisting modules. Janssen and Lindenmayer [14] (see also [27, Chapter 3] and [28]) showed that context-free models are too weak to capture the whole spectrum of developmental sequences in plants. For example, the *basipetal* flowering sequence observed in many compound inflorescences requires the use of one or more signals that propagate through the developing structure and control the opening of buds. Such a sequence is characterized by the first flower opening at the top of the main axis and the flowering zone progressing downward towards the base of the plant.

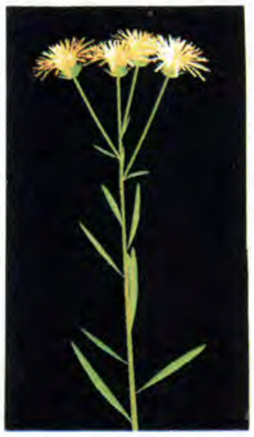


Figure 14: A model of *Hieracium umbellatum*.

Figure 14 shows a synthetic image of *Hieracium umbellatum*, a sample composite plant with a basipetal flowering sequence. Following *model I* postulated by Janssen and Lindenmayer, we assume that the opening of buds is controlled by a *hormone* generated at some point of time near the base of the plant and transported towards the apices. The hormone propagates faster in the main axis than in the lateral branches. As a result, it first reaches the bud of the main axis, then those of the lateral branches in the basipetal sequence. The growth of the main axis and of the lateral branches stops when the hormone attains their respective terminal buds. In addition, the hormone penetrating

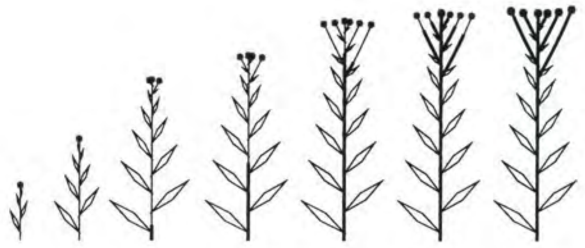


Figure 15: Development of *Hieracium umbellatum*. The stages shown represent frames 170, 265, 360, 400, 470, 496, and 520 of an animated sequence.

a node stops the development of a leaf originating at this node. Snapshots from a diagrammatic animated developmental sequence illustrating this process are shown in Figure 15.

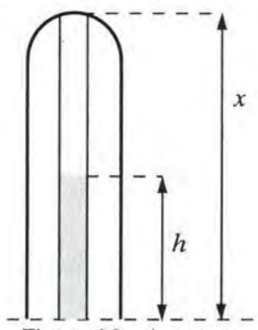


Figure 16: A conceptual model of the apex.

The complete listing of the dL-system capturing the development of *Hieracium* is too long to be included in this paper, but a specification of the activities of the main apex provides a good illustration of the context-sensitive control mechanism involved. We conceptualize this apex as a growing and periodically dividing tube of length x, which may be penetrated by the hormone to a height $h \le x$ (Figure 16). The apex can assume three states: F_{a0} (not yet reached by the hormone), F_{a1} (being penetrated by the hormone), and F_{a2} (completely filled with the hormone). The apical behavior is captured by the following rules:

```
$F_l(x_l, h_l) < F_{a0}(x)$ :
    **if** $x_l > h_l$ & $x < x_{th}$
        **solve** $\frac{dx}{dt} = G(x)$
    **if** $x = x_{th}$
        **produce** $F_{i0}(kx)[F_{a0}(0)]F_{a0}((1-k)x)$
    **if** $x_l = h_l$ & $x < x_{th}$
        **produce** $F_{a1}(x, 0)$
$F_{a1}(x, h)$:
    **if** $x > h$ & $x < x_{th}$
        **solve** $\frac{dx}{dt} = G(x)$, $\frac{dh}{dt} = v$
    **if** $kx > h$ & $x = x_{th}$
        **produce** $F_{i1}(kx, h)[F_{a0}(0)]F_{a0}((1-k)x)$
    **if** $x > h \ge kx$ & $x = x_{th}$
        **produce** $F_{i2}(kx, kx)F_{a1}((1-k)x, h-kx)$
    **if** $x = h$
        **produce** $F_{a2}(x, x)$
```

The first three rules model the apex without the hormone. If the preceding internode F_l is not yet completely penetrated by the hormone ($x_l > h_l$) and the length x of the apex is below the threshold value x_{th}, the apex elongates according to the growth function $G(x)$. Upon reaching the threshold length ($x = x_{th}$), the apex F_{a0} subdivides, producing an internode F_{i0} and a lateral apex F_{a0}. Finally, once the hormone penetrates the entire internode F_l (as indicated by the condition $x_l = h_l$), it flows into the apex, which then changes its state to F_{a1}.

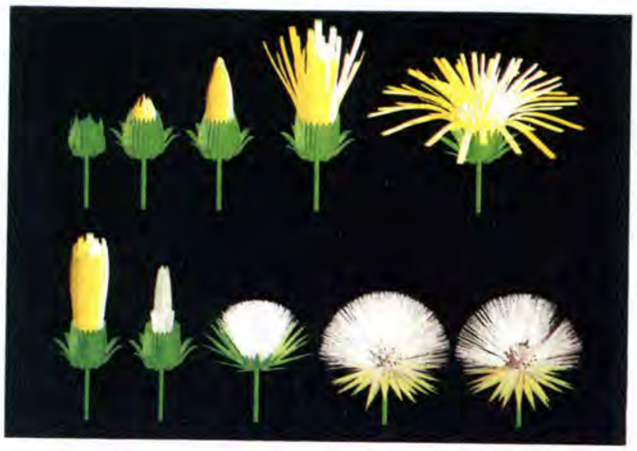


Figure 17: Development of a single flower head of *Hieracium umbellatum*

The continuous rule for F_{a1} describes the growth of the apex with rate $G(x)$ and the propagation of the hormone with constant speed v. The next two productions capture the alternate cases of the apex subdivision, with the hormone level h below or above the level kx at which the new internode splits from the apex. The last production is applied when the hormone reaches the tip of the apex, and changes its state to the flowering state F_{a2}.

The complete model of *Hieracium umbellatum* contains additional rules that describe the elongation of internodes, the propagation of the hormone within and between the internodes, and the development of flower heads. The heads undergo the sequence of transformations illustrated in Figure 17. The bracts (green parts of the flower head) have been represented using Bézier patches controlled by the dL-system, while the petals have been formed as extending chains of filled rectangles, with the angles between consecutive rectangles controlled by cubic growth functions. This technique allowed us to represent each petal with a relatively modest number of polygons (10).

8 CONCLUSIONS

We have introduced differential L-systems as a combined discrete/continuous model suitable for computer simulation and animation of plant development. Continuous aspects of module behavior are described by ordinary differential equations, and discontinuous qualitative changes are captured by productions. The link between L-systems and dL-systems makes it possible to use existing discrete developmental models as a starting point for constructing dL-systems suitable for animation.

Differential L-systems have a wide spectrum of prospective applications, ranging from modest projects, such as the diagrammatic animation of developmental mechanisms employed by plants, to ambitious ones, such as the realistic animation of the growth of extinct plants. On the conceptual level, dL-systems expand piecewise-continuous differential equations with a formal specification of discrete changes to system configuration. The resulting formalism makes it possible to model developing branching structures with a theoretically unlimited number of modules. From a different perspective, dL-systems can be considered as the continuous-time extension of parametric L-systems.

The following problems still require solutions:

- **Combined differential-algebraic specification of continuous processes.** In some cases it is convenient to describe continuous aspects of model behavior using explicit functions of time instead of differential equations. For example, the expression of the cubic growth function using the differential equation presented in Section 6 is somewhat artificial. In order to accommodate explicit function specifications, the definition of dL-systems should be extended to comprehend differential-algebraic equations.
- **Incorporation of stochastic rules.** Differential L-systems have been formulated in deterministic terms. Stochastic rules should be incorporated to capture the specimen-to-specimen variations in modeled plants, as has been done for L-systems.
- **Development of the simulation software.** The simulations discussed in this paper were carried out using a programming language based on parametric L-systems [13, 26]. In this environment, the user must explicitly specify the formulae for numerically solving the differential equations included in the models (the forward Euler method was used in all cases). From the user's perspective, it would be preferable to incorporate a differential equation solver into the simulator, and specify the models directly in terms of dL-systems.
- **Improved realism of dL-system models.** We have not addressed many practical problems related to the construction of realistic models, such as the avoidance of intersections between modules, the improved modeling of growing plant organs (petals, leaves, and fruits), and the simulation of wilting.

The simulation and visualization of natural phenomena has the intriguing charm of blurring the line dividing the synthesis of images from the re-creation of nature. The animation of plant development adds a new phenomenon to this (un)real world.

Acknowledgements

We would like to thank Jim Hanan for his essential work on the plant modeling software *cpfg* used in the simulations, and for valuable references and comments. At different points in time, Gavin Miller, Karl Sims and Alvy Ray Smith revealed to us the techniques used in their developmental animations. M. Raju and C. C. Chinnappa explained the details of the development of *Lychnis coronaria* and *Hieracium umbellatum*. We also gained many insights from illuminating discussions with Bill Remphrey, John Reinitz, Stan Letovsky, and Keith Ferguson. This research was sponsored by an operating grant and a graduate scholarship from the Natural Sciences and Engineering Research Council of Canada, and a grant from the U.S. Air Force Office of Scientific Research.

References

[1] M. Aono and T. L. Kunii. Botanical tree image generation. *IEEE Computer Graphics and Applications*, 4(5):10–34, 1984.

[2] R. Barzel. *Physically-based modeling for computer graphics — a structured approach.* Academic Press, Boston, 1992.

[3] C. G. de Koster and A. Lindenmayer. Discrete and continuous models for heterocyst differentiation in growing filaments of blue-green bacteria. *Acta Biotheoretica*, 36:249–273, 1987.

[4] P. de Reffye, C. Edelin, J. Françon, M. Jaeger, and C. Puech. Plant models faithful to botanical structure and development. Proceedings of SIGGRAPH '88 (Atlanta, Georgia, August 1–5, 1988), in *Computer Graphics* 22, 4 (August 1988), pages 151–158, ACM SIGGRAPH, New York, 1988.

[5] L. Edelstein-Keshet. *Mathematical models in biology.* Random House, New York, 1988.

[6] D. A. Fahrland. Combined discrete event – continuous systems simulation. *Simulation*, 14(2):61–72, 1970.

[7] K. W. Fleischer and A. H. Barr. A simulation testbed for the study of multicellular development: Multiple mechanisms of morphogenesis. To appear in *Artificial Life III*, Addison-Wesley, Redwood City, 1993.

[8] J. D. Foley, A. van Dam, S. Feiner, and J. Hughes. *Computer graphics: Principles and practice.* Addison-Wesley, Reading, 1990.

[9] F. D. Fracchia, P. Prusinkiewicz, and M. J. M. de Boer. Animation of the development of multicellular structures. In N. Magnenat-Thalmann and D. Thalmann, editors, *Computer Animation '90*, pages 3–18, Tokyo, 1990. Springer-Verlag.

[10] J. Françon. Sur la modélisation de l'architecture et du développement des végétaux. In C. Edelin, editor, *L'Arbre. Biologie et Développement.* Naturalia Monspeliensia, 1991. N̲o hors série.

[11] N. Greene. Organic architecture. SIGGRAPH Video Review 38, segment 16, ACM SIGGRAPH, New York, 1988.

[12] N. Greene. Voxel space automata: Modeling with stochastic growth processes in voxel space. Proceedings of SIGGRAPH '89 (Boston, Mass., July 31–August 4, 1989), in *Computer Graphics* 23, 4 (August 1989), pages 175–184, ACM SIGGRAPH, New York, 1989.

[13] J. S. Hanan. *Parametric L-systems and their application to the modelling and visualization of plants.* PhD thesis, University of Regina, June 1992.

[14] J. M. Janssen and A. Lindenmayer. Models for the control of branch positions and flowering sequences of capitula in *Mycelis muralis* (L.) Dumont (Compositae). *New Phytologist*, 105:191–220, 1987.

[15] W. Kreutzer. *System simulation: Programming styles and languages.* Addison-Wesley, Sydney, 1986.

[16] A. Lindenmayer. Mathematical models for cellular interaction in development, Parts I and II. *Journal of Theoretical Biology*, 18:280–315, 1968.

[17] A. Lindenmayer and H. Jürgensen. Grammars of development: Discrete-state models for growth, differentiation and gene expression in modular organisms. In G. Rozenberg and A. Salomaa, editors, *Lindenmayer systems: Impacts on theoretical computer science, computer graphics, and developmental biology*, pages 3–21. Springer-Verlag, Berlin, 1992.

[18] B. B. Mandelbrot. *The fractal geometry of nature.* W. H. Freeman, San Francisco, 1982.

[19] G. S. P. Miller. Natural phenomena: My first tree. Siggraph 1988 Film and Video Show.

[20] G. J. Mitchison and M. Wilcox. Rules governing cell division in *Anabaena. Nature*, 239:110–111, 1972.

[21] E. Mjolsness, D. H. Sharp, and J. Reinitz. A connectionist model of development. *Journal of Theoretical Biology*, 152(4):429–454, 1991.

[22] H. Noser, D. Thalmann, and R. Turner. Animation based on the interaction of L-systems with vector force fields. In T. L. Kunii, editor, *Visual computing – integrating computer graphics with computer vision*, pages 747–761. Springer-Verlag, Tokyo, 1992.

[23] P. Prusinkiewicz. Graphical applications of L-systems. In *Proceedings of Graphics Interface '86 — Vision Interface '86*, pages 247–253, 1986.

[24] P. Prusinkiewicz, M. Hammel, and J. Hanan. Lychnis coronaria. QuickTime movie included in the Virtual Museum CD-ROM, Apple Computer, Cupertino, 1992.

[25] P. Prusinkiewicz and J. Hanan. Visualization of botanical structures and processes using parametric L-systems. In D. Thalmann, editor, *Scientific Visualization and Graphics Simulation*, pages 183–201. J. Wiley & Sons, Chichester, 1990.

[26] P. Prusinkiewicz and J. Hanan. L-systems: From formalism to programming languages. In G. Rozenberg and A. Salomaa, editors, *Lindenmayer systems: Impacts on theoretical computer science, computer graphics, and developmental biology*, pages 193–211. Springer-Verlag, Berlin, 1992.

[27] P. Prusinkiewicz and A. Lindenmayer. *The algorithmic beauty of plants.* Springer-Verlag, New York, 1990. With J. S. Hanan, F. D. Fracchia, D. R. Fowler, M. J. M. de Boer, and L. Mercer.

[28] P. Prusinkiewicz, A. Lindenmayer, and J. Hanan. Developmental models of herbaceous plants for computer imagery purposes. Proceedings of SIGGRAPH '88 (Atlanta, Georgia, August 1–5, 1988), in *Computer Graphics* 22, 4 (August 1988), pages 141–150, ACM SIGGRAPH, New York, 1988.

[29] L. F. Shampine, I. Gladwell, and R. W. Brankin. Reliable solution of special event location problems for ODEs. *ACM Transactions on Mathematical Software*, 17, No. 1:11–25, March 1991.

[30] K. Sims. Panspermia. SIGGRAPH Video Review, ACM SIGGRAPH, New York, 1990.

[31] A. R. Smith. Plants, fractals, and formal languages. Proceedings of SIGGRAPH '84 (Minneapolis, Minnesota, July 22–27, 1984) in *Computer Graphics*, 18, 3 (July 1984), pages 1–10, ACM SIGGRAPH, New York, 1984.

Modeling Soil: Realtime Dynamic Models for Soil Slippage and Manipulation

Xin Li and J. Michael Moshell

Institute for Simulation and Training
University of Central Florida

ABSTRACT
A physically based model of an object is a mathematical representation of its behavior, which incorporates principles of Newtonian physics. Dynamic soil models are required in animations and realtime interactive simulations in which changes of natural terrain are involved. Analytic methods, based on soil properties and Newtonian physics, are presented in the paper to model soil slippage and soil manipulations. These methods can be used to calculate the evolution of a given soil configuration under the constraint of volume conservation and to simulate excavating activities such as digging, cutting, piling, carrying or dumping soil. Numerical algorithms with linear time and space complexities are also developed to meet the requirement of realtime computer simulation.

CR Categories: I.3.7 [Computer Graphics]: Graphics and Realism; I.6.3 [Simulation and Modeling]: applications.
Additional Keywords: physically based modeling, realtime simulation, soil dynamics, slippage, soil manipulation.

1. INTRODUCTION

Physically-based modeling is a growing area of computer graphics research. A good deal of work has been done toward physically based models of objects such as rigid and nonrigid bodies, hydraulic surfaces or natural terrain. However, soil models that are both physically realistic and computationally efficient in realtime simulations have not been developed. Recently, substantial interest in dynamic soil models has been expressed by some developers of realtime simulations of Dynamic Terrain systems. Such systems provide the capability, within a realtime graphical simulation, of reconstructing landscape architecture or rearranging the terrain surface. These systems essentially involve allowing the simulation's user to conduct excavating activities in the terrain database at any freely chosen location. These activities may include digging ditches, piling up dirt, cutting the soil mass from the ground, carrying it for a distance, and dumping it at another location. To these deformations, the soil mass must behave in realistic manners under external stimuli.

Moshell and Li developed a visually plausible kinematic soil model [10]. In their work, a bulldozer blade serves as a local force function used to change the heights of the terrain. Excess terrain volume which is "scaped off" by the moving blade is added to the moving berm in front of the blade. The berm is then smoothed by a bidirectional Cardinal spline algorithm. The demonstration of the model appears realistic and runs in realtime. The simulation, however, is kinematic. No forces are computed. The soil does not slump when the bulldozer leaves. The volume of given soil is not conserved.

Burg and Moshell focussed on the problem of piling up soil such that the soil spills down from the mounds in a realistic-looking way [3]. In their approach, the terrain is modeled by a 2-d grid of altitude posts. Constraint equations are defined to describe relationships among altitude posts and their neighbors. An iterative relaxation algorithm, suggested in [11], is used to simulate the falling soil. The constraints enforce an averaging or "smoothing" of each altitude post with its neighbors. The algorithm is volume-preserving under certain conditions. The model is purely kinematic. The physical properties of different types of soil are not modeled.

Our research work is focused on dynamic models of soil slippage and soil manipulations. For the slippage model, we determine if a given soil configuration is in static equilibrium, calculate forces which drive a portion of the soil to slide if the configuration is not stable, and meanwhile preserve the volume conservation. For the soil manipulation models, we investigate interactions between soil and excavating machines, implement a bulldozer model and a scooploader model. These models are based on analytic methods and Newtonian physics. The computational times of the corresponding algorithms are fast enough to meet the requirement of realtime graphical simulations. For clarity, this paper mainly focuses on the 2-d case. Extensions to 3-d have been completed and are briefly discussed.

2. PRELIMINARIES

The discussion of soil models needs some understanding of soil properties. In this section, we introduce some concepts which are borrowed directly from civil engineering. Interested readers are referred to [2], [4], [5] and [7] for more details.

The *shear strength* of the soil is the resistance per unit area to deformation by continuous shear displacement of soil particles along surfaces of rupture. It may be attributed to three basic components: 1) frictional resistance to sliding among soil particles; 2) cohesion and adhesion among soil particles; and 3) interlocking of solid particles to resist deformation. (Cohesion is molecular attraction among like particles. Adhesion is a molecular attraction among unlike particles.)

The *shear stress*, on the other hand, is the force per unit area experienced by a slope, which pushes the mass to move along the failure plane. The combined effects of gravity and water are the primary influences on the shear stress. It may also be influenced by some natural phenomena such as chemical actions, earthquakes, or wind.

The shear *strength force* and *stress force*, denoted by s and τ respectively, are defined as the shear strength and stress multiplied by the total area. The measure of s and τ can be determined from the *Mohr-Coulomb theory* indicated in [5]:

12424 Research Parkway, Suite 300, Orlando, Florida.
Phone: (407) 658-5073, email: lix@ist.ucf.edu

This work is sponsored by the U.S. Army Simulation, Training and Instrumentation Command (STRICOM). All opinions are, however, solely due to the authors.

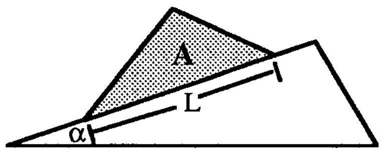


Fig. 1: The Failure Plane

(2.1) $s = c\,L + W\cos(\alpha)\tan(\phi)$

(2.2) $\tau = W\sin(\alpha)$

where L is the length of the failure plane, α is the degree of natural slope, and $W=\gamma A$ is the weight of soil above the failure plane (see Fig. 1). c, ϕ and γ describe properties of the soil, where c indicates the cohesion, ϕ is the angle of internal friction (i.e. it is a measure of the friction among soil particles) and γ is the unit weight. Some typical parameters and their units are listed in the table below [1]:

SOIL TYPE	c (t/m)	ϕ (degree)	γ (t/m^2)
dry sand	0	26-33	1.9-2.0
Sandy loam	0-2.0	14-26	1.8-2.0
Loam	0.5-5.0	10-28	1.8-2.1

Soil is a very complex material. It may be influenced by changes in the moisture content, pore pressures, structural disturbance, fluctuation in the ground water table, underground water movements, stress history, time, chemical action or environmental conditions. Predicting the changes of complex configurations is either intractable or highly costly. However, for many interactive applications, speed and realistic appearance are more important than accuracy. Hence in this paper, we assume that only homogeneous and isotropic soil will be processed. Conditions such as seepage, pore pressure, existence of tension cracks and deformation resulting from permanent atomic dislocation will not be considered.

3. STATIC EQUILIBRIUM AND RESTORING FORCE

In this section, we develop methods to determine whether or not a given configuration is stable, calculate the critical angle above which sliding occurs, and quantify the force which pushes the soil mass moving along the failure plane.

3.1 STABILITY

The stability of a given soil configuration is determined by the *factor of safety*, denoted by F, of a potential failure surface. From the Mohr-Coulomb theory, F is defined as a ratio between the strength force and the stress force [5]:

(3.1) $$F = \frac{s}{\tau} = \frac{c\,L + W\cos(\alpha)\tan(\phi)}{W\sin(\alpha)}$$

When F is greater than 1, the configuration is said to be in a state of equilibrium. Otherwise, failure is imminent. To analyze the factor of safety, we divide the given soil mass into n slices with equal width Δx:

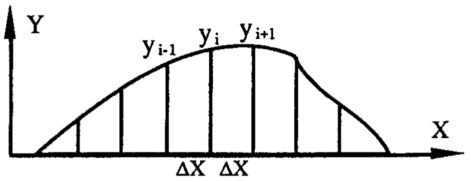


Fig. 2: Dividing the given mass into small slices

The calculation of the factor of safety of each slice can be done individually. The following free body diagram shows forces applied on slice i:

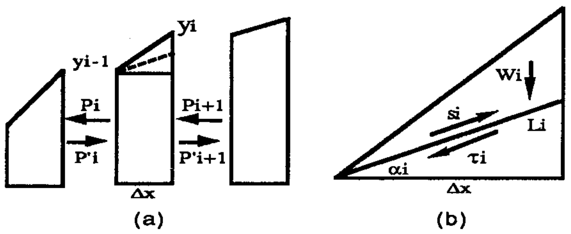


Fig. 3: Free body diagram for slice i

In (a), the P's are forces exerted between slices. They are pairwise equal and in opposite directions and thus can be cancelled. At any time t, therefore, sliding can only happen in the top triangle area of a slice. (b) shows forces acting on this area, where strength and stress forces are given by (2.1) and (2.2) with L, W and α replaced by L_i, W_i and α_i respectively.

To determine if there exists a failure angle α_i (so that the soil mass above it will slide) and calculate the net force exerted on the failure plane if α_i does exist, we start from (3.1). Note that L_i and W_i can be expressed in terms of α_i. Replacing L_i and W_i in (3.1) with functions of α_i, we obtain

(3.2) $$F(\alpha_i) = \frac{2c+\gamma\tan(\phi)[h\cos(\alpha_i)-\Delta x\sin(\alpha_i)]\cos(\alpha_i)}{\gamma(h\cos(\alpha_i)-\Delta x\sin(\alpha_i))\sin(\alpha_i)}$$

where $h=y_i-y_{i-1}$ is the height of the triangle in Fig. 3-(b). For any angle $\alpha_i>\tan^{-1}(h/\Delta x)$, function $F(\alpha_i)$ makes no physical sense. In the range of $[0, \tan^{-1}(h/\Delta x)]$, $F(\alpha_i)$ reaches its minimum when the first derivative of $F(\alpha_i)$, with respect to α_i, is equal to 0. That is

(3.3) $$\frac{dF}{d\alpha} = \frac{1}{\tau^2}\,[A\cos(2\alpha_i) + B\sin(2\alpha_i) + C] = 0$$

where

$$A = \frac{\gamma^2}{2}\tan(\phi)(\Delta x^2-h^2)-2\gamma ch,$$

$$B = \gamma^2 h\Delta x\tan(\phi)+2\gamma c\Delta x, \text{ and}$$

$$C = -\frac{\gamma^2}{2}\tan(\phi)(\Delta x^2+h^2).$$

Solving (3.3) gives us four angles (see [9]). We can choose the one which satisfies $0\le\alpha_i\le\tan^{-1}(h/\Delta x)$ in (3.2) to calculate the factor of safety F. The given configuration is statically stable if F>1. Otherwise sliding is inevitable.

3.2 CRITICAL SLOPE ANGLE

Suppose that we have F<1 for a given configuration. In the range of $[0, \tan^{-1}(h/\Delta x)]$ there are at most two angles, say β_1 and β_2, such that $F(\beta_1)=F(\beta_2)=1$. The angle $\beta_0=\min(\beta_1, \beta_2)$ is said to be the *critical-slope angle* of the configuration. Above this angle impending slip occurs. β_1 and β_2 can be obtained by solving the equation (3.4) for α:

(3.4) $$F(\alpha) = \frac{2c+\gamma\tan(\phi)[h\cos(\alpha)-\Delta x\sin(\alpha)]\cos(\alpha)}{\gamma[h\cos(\alpha)-\Delta x\sin(\alpha)]\sin(\alpha)} = 1$$

where all symbols are as explained earlier. The solution to (3.4) is derived in [9].

3.3 RESTORING FORCE

Let a configuration be given in Fig.4-(a) with β_0 as the critical-slope angle. The force that pushes the mass in the triangle along the edge gh_0 can be computed as follows. First the line segment h_0h_n is divided into n small segments with equal length Δh. Fig 4-(b) shows the free body diagram of the i-th dovetail indicated by the shaded area in (a).

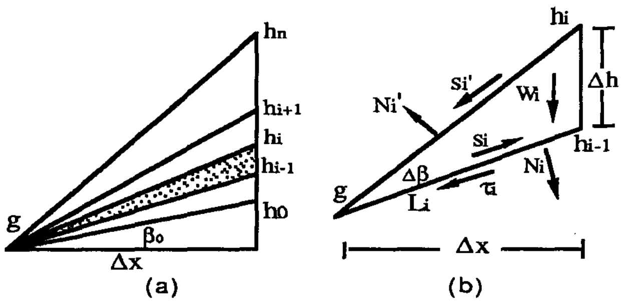


Fig. 4: Analyzing the restoring force

Let's analyze forces exerted on the dovetail. The weight W_i can be decomposed into two forces, namely N_i and τ_i, which are normal and parallel to the edge L_i respectively. s_i is the strength force resisting the sliding motion, s_i' the opponent force generated by strength force s_{i+1}, and N_i' the force supporting the dovetail above it. The net force f_i applied on dovetail-i is therefore given by a vectorial summation:

$$(3.5) \qquad f_i = N_i + \tau_i + s_i + s_i' + N_i'$$

The total net force f acting on the whole triangle area is the summation of f_i's, $1 \le i \le n$, i.e.

$$(3.6) \qquad f = \sum_{i=1}^{n}(N_i+\tau_i+s_i+s_i'+N_i') = \sum_{i=2}^{n}\tau_i$$

since $\tau_1 = s_1$ (due to $F(\beta_0)=1$), $N_n'=0$, $s_n'=0$, $N_i' = N_{i+1}$ and $s_i' = -s_{i+1}$ for $1 \le i \le n-1$. Based on (3.6) and Fig.4, [9] gives a derivation of (3.7) by letting Δh tend to zero.

$$(3.7) \qquad f = \frac{\gamma\Delta x^2}{4} \mathrm{Ln}\left(\frac{h_n^2+\Delta x^2}{h_0^2+\Delta x^2}\right)\cos(\beta_0) + \frac{\gamma\Delta x}{2}(h_n - h_0 - \Delta x(\beta_n - \beta_0))\sin(\beta_0)$$

where $\beta_n = \tan^{-1}(h_n/\Delta x)$ and $\beta_0 = \tan^{-1}(h_0/\Delta x)$. (3.7) can be used to quantify the total force on the top triangle area of each slice.

4. VOLUME CONSERVATION

The approach used in this section is strongly related to [8]. Recall that, in the previous discussion, a given configuration is divided into n slices. The i-th slice, $1 \le i \le n$, can be conveniently thought of as a container holding an amount of soil whose quantity is given by $(y_i+y_{i-1})\Delta x/2$.

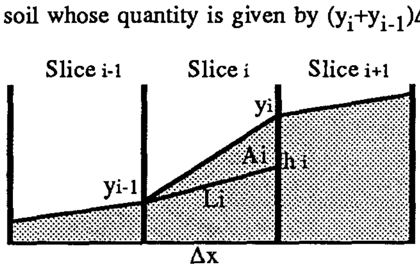


Fig. 5: Considering slices as containers

Let us consider a small change, denoted by ΔW_i, of the mass W_i in slice$_i$. Since $W_i = (y_i+y_{i-1})\gamma\Delta x/2$, we have

$$(4.1) \qquad \Delta W_i = (y_i+\Delta y_i+y_{i-1}+\Delta y_{i-1})\gamma\Delta x/2 - (y_i+y_{i-1})\gamma\Delta x/2 = (\Delta y_i+\Delta y_{i-1})\gamma\Delta x/2$$

On the other hand, let us assume that there is a force f_i exerted on the triangle area A_i at the top of slice$_i$, which is parallel to the edge L_i. Due to f_i, A_i tends to move along the direction of f_i at a velocity v_i. The rate of the "flow" of mass of A_i through slice i can be computed by $\gamma A_i v_i/\Delta x$. Thus, the "mass throughput" of slice$_i$ can be quantified by $\gamma A_i v_i \Delta t/\Delta x$, where Δt is a unit of time. Similarly, the mass throughput of slice$_{i+1}$ is given by $\gamma A_{i+1}v_{i+1}\Delta t/\Delta x$.

From the principle of volume conservation, the change of soil quantity in slice$_i$ is the amount of soil which goes out, minus the amount of soil which goes in. It can be expressed by

$$(4.2) \qquad \Delta W_i = \frac{\gamma A_i}{\Delta x} v_i \Delta t - \frac{\gamma A_{i+1}}{\Delta x v_{i+1}} \Delta t$$

where $A_i = (y_i - h_i)\Delta x/2$. Putting (4.1) and (4.2) together and rearranging it, we have

$$(4.3) \qquad \frac{\Delta y_i}{\Delta t} + \frac{\Delta y_{i-1}}{\Delta t} = \frac{1}{\Delta x}[(y_i-h_i))v_i - (y_{i+1}-h_{i+1}))v_{i+1}]$$

Now let Δt tend to 0. It follows that

$$(4.4) \qquad \frac{dy_i}{dt} + \frac{dy_{i-1}}{dt} = \frac{1}{\Delta x}[(y_i-h_i))v_i - (y_{i+1}-h_{i+1}))v_{i+1}]$$

Recall that (3.7) gives us a formula to compute force f_i. From Newton's second law, we have

$$(4.5) \qquad f_i = \gamma A_i \frac{dv_i}{dt} = \frac{\gamma\Delta x}{2}(y_i-h_i)\frac{dv_i}{dt}$$

Rearranging, we obtain both

$$(4.6) \qquad \frac{dv_i}{dt} = \frac{2f_i}{\gamma\Delta x(y_i-h_i)}, \text{ and}$$

$$(4.7) \qquad v_i = \frac{2}{\gamma\Delta x}\int \frac{f_i}{y_i-h_i}dt$$

Now we take the second derivative of (4.4) with respect to t and plug (4.6) and (4.7) into the resulting formula. That yields

$$(4.8) \qquad \frac{d^2y_i}{dt^2} + \frac{d^2y_{i-1}}{dt^2} = \frac{2}{\gamma\Delta x}\left[\frac{dy_i-dh_i}{dt}\int\frac{f_i}{y_i-h_i}dt + f_i - \frac{dy_{i+1}-dh_{i+1}}{dt}\int\frac{f_{i+1}}{y_{i+1}-h_{i+1}}dt + f_{i+1}\right]$$

Note that we can denote h_i and f_i as functions of y_{i-1} and y_i, i.e. $h_i = h(y_{i-1}, y_i)$ and $f_i = f(y_{i-1}, y_i)$, since they can be determined based only on y_{i-1} and y_i if Δx and other soil properties are fixed. Hence, (4.8) is an equation with three variables, namely y_{i-1}, y_i, y_{i+1}. Let us suppose that we have divided the given configuration into n slices. Now we end up with n+1 unknowns, $y_0, y_1, \ldots, y_n$, and n+1 ordinary differential equations involving y_i's, their time derivatives and integrals. Solving these equations, we will obtain the solution for the soil behavior which satisfies both the soil dynamics and the volume conservation.

5. NUMERICAL SOLUTION

In this section we linearize equations (4.8) for both purposes of simplification and discretization. we start from (4.4). Assume that, at any instance of time t_m, velocity v_i of the mass on the top of slice$_i$ is represented by $v_i(t_m)$, the value of y_i is represented by $y_i(t_m)$, the rate of the change of y_i is represented by $y_i'(t_m)=dy_i(t_m)/dt$. Then, at the very next time instance t_{m+1}, the force $f_i=f_i(y_{i-1}(t_m), y_i(t_m))$ can be computed by (3.7) according to the value of y_{i-1} and y_i from the previous step. If the Euler integration algorithm is used, the velocity v_i at the time t_{m+1} can be computed by

$$(5.1) \qquad v_i(t_{m+1}) = v_i(t_m) + \frac{f_i(y_{i-1}(t_m), y_i(t_m))}{W_i}\Delta t$$

where Δt is the integration step size. Similarly $v_{i+1}(t_{m+1})$ is calculated. It follows that, from (4.5), we have

$$(5.2) \qquad y_i'(t_{m+1}) + y_{i-1}'(t_{m+1})$$
$$= \frac{1}{\Delta x}[(y_i(t_m)-h(y_{i-1}(t_m),y_i(t_m)))v_i(t_{m+1})$$
$$- (y_{i+1}(t_m)-h(y_i(t_m),y_{i+1}(t_m)))v_{i+1}(t_{m+1})]$$

Since at the time instance t_{m+1}, all items on the right hand side are knowns, either from the previous step of the simulation or from the calculations of $v_i(t_{m+1})$ and $v_{i+1}(t_{m+1})$, we may treat it as a constant, namely C_i. We now have n+1 equations in the following format:

$$(5.3) \qquad \begin{aligned} &y_0'(t_{m+1}) = C_0 \\ &y_1'(t_{m+1}) + y_0'(t_{m+1}) = C_1 \\ &\dots\dots \\ &y_n'(t_{m+1}) + y_{n-1}'(t_{m+1}) = C_n \end{aligned}$$

Solving (5.3) for $y_i'(t_{m+1})$, i=0, 1, ... n, we will be able to use the Euler method again to determine the new values for each y_i:

$$(5.4) \qquad y_i(t_{m+1}) = y_i(t_m) + y_i'(t_{m+1})\Delta t$$

Algorithm 1 describes the procedure of the numerical solution, in which each step of the algorithm takes linear time to execute. Thus the time complexity of the algorithm is O(n) where n is the number of elevation posts in a given configuration. The space required to store forces, velocities and heights of posts is also proportional to n.

Algorithm 1.

```
At any time $t_{m+1}$ of simulation, do the following:
1) for each post $y_i$, calculate its mass velocity $v_i(t_{m+1})$ by
   using (5.1);
2) for $y_i$, compute the right hand side of (5.2);
3) use forward substitution to solve equations (5.3) for
   $y_i'(t_{m+1})$, i=0, 1, ... n;
4) use Euler integration to determine new value for each
   $y_i(t_{m+1})$.
```

6. EXTENSION TO 3-D

In going to 3-d soil dynamics, we use some essential concepts and results from the discussion on 2-d. First, a given soil configuration is partitioned into small prisms. The values of elevation posts (i.e. vertices) of each prism are evolved by an approximation procedure as follows.

Consider, in Fig. 6, the post z(i,j) chosen arbitrarily:

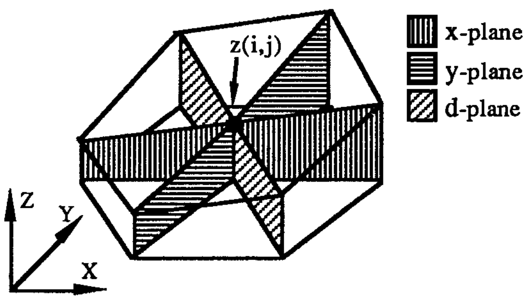


Fig. 6: An approximation of the 3-d configuration

z(i, j) is surrounded by six prisms. At any time instance t, those prisms are the only ones that affect the height of z(i,j). The effect caused by those prisms can be approximated by considering forces exerted on three planes, namely the x-plane, y-plane and d-plane. They are indicated by different types of shaded areas in Fig. 6. Thus the 3-d problem is reduced to a 2-d problem. The finer the partitioning is, the smaller the base triangles of prisms are, and the more accurate the approximation will be.

Let's assume that, at any time t_m, the height of post z(i,j) is represented by $z_{ij}(t_m)$, and the rate of change of z(i,j) is represented by $z_{ij}'(t_m)$. Since $z_{ij}'(t_m)$ is affected by forces from 3 planes, it can be expressed as a summation of three terms:

$$(6.1) \quad z_{ij}'(t_m) = zx_{ij}'(t_m) + zy_{ij}'(t_m) + zd_{ij}'(t_m)$$

where $zx_{ij}'(t_m)$, $zy_{ij}'(t_m)$ and $zd_{ij}'(t_m)$, are rates of changes of $z_{ij}'(t_m)$ caused by forces exerted on the x-plane, y-plane and d-plane respectively.

During a simulation, each time slice Δt is divided into two substeps $\Delta t1$ and $\Delta t2$. In $\Delta t1$, we first use (3.7) to compute forces exerted on three different planes. Then $zx_{ij}'(t_{m+1})$, $zy_{ij}'(t_{m+1})$ and $zd_{ij}'(t_{m+1})$ can be obtained by solving equations (5.3). In step $\Delta t2$, Euler integration is used to determine new values for each $z_{ij}(t_{m+1})$:

$$(6.2) \quad z_{ij}(t_{m+1}) = z_{ij}(t_m) + [zx_{ij}'(t_m) + zy_{ij}'(t_m) + zd_{ij}'(t_m)]\Delta t$$

For $\Delta t1$ and $\Delta t2$ of each iteration in the simulation, we split our 2-d computational problem into 3 terms: x-plane scan, y-plane scan and d-plane scan. Each scan has two phases corresponding to two time substeps. A scan on any plane involves calculations of forces exerted on that plane, rates of changes of z(i, j) caused by the forces, new height of each post, etc. Computations for each scan in a time substep are independent of scans on the other planes in the same substep, and therefore can be performed either sequentially or in parallel. It is important to notice that, in the same time substep, scans in different orders (x-scan then y-scan then d-scan, or y-scan then x-scan then d-scan, etc.) will have the same effect. The reasons are discussed in [9].

The 3-d algorithm can be briefly described as follows: Each iteration of simulation is divided into two phases. Steps (1)-(3) of Algorithm 1 are performed first for each scan. Then step (4) is applied for each scan to calculate new values of posts. Both time and space complexity of the 3-d algorithm remain linear in the number of posts.

7. INTERACTION BETWEEN SOIL AND BLADE

In this section, we analyze the interaction between the soil mass and a bulldozer's blade. Let's assume that the height of the blade is H. The shape of the blade can be modeled by an arc of a circle centered at the location $<x_c,y_c>$ with radius R. We divide the arc into n segments, each of which has length $R\Delta\beta$. Furthermore, the soil mass in front of the blade is also partitioned into n slices by horizontal lines at each joint point of two arc segments as shown in Fig. 7.

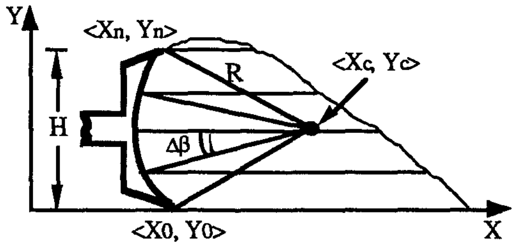


Fig. 7: Dividing the blade and soil mass

To calculate the force resisting cutting, we arbitrarily pick the i-th slice from the partitioning. The arc segment can be approximated by a line segment from point $<x_i,y_i>$ to point $<x_{i+1}, y_{i+1}>$. Note that the length of the line segment, denoted by ΔL, approaches the length of the arc when $\Delta\beta$ approaches 0. The idea is explained in Fig. 8:

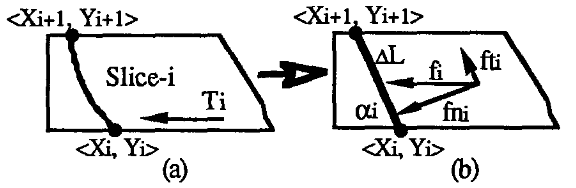


Fig. 8: Free body diagram for i-th slice

If the cutting part of the bulldozer pushes the soil mass with enough force, the equilibrium will be destroyed. At this moment, the resistance parallel to blade motion at the point $<x_i,y_i>$ can be calculated by the formula [1]:

$$(7.1) \qquad T_i = Ae^{2\alpha_i+B}[\gamma(H + y_0 - y_i) + c\cot(\phi)]\tan(\phi)$$

where T_i is the localized shear stress and α_i is the magnitude of the angle of inclination of ΔL to the horizon. The remaining symbols are as explained earlier. All angles are given in radians. Constants A and B are only related to ϕ and δ (δ is the angle of external friction), of the given soil:

$$(7.2) \qquad A = \frac{\sin(\delta)\,[\,\cos(\delta) + \sqrt{\sin^2(\phi) - \sin^2(\delta)}\,]}{1 - \sin(\phi)}$$

$$(7.3) \qquad B = \delta + \sin^{-1}\left(\frac{\sin(\delta)}{\sin(\phi)}\right) - \pi$$

Due to different cutting depths (given by $H+y_0-y_i$) and different inclination angles α_i, the magnitudes of T_i vary. The resistance force exerted on ΔL can be computed by $f_i = T_i\,\Delta L$. As shown in Fig. 8-(b), f_i can be further decomposed into two components, one normal to ΔL and another parallel to ΔL. The normal force is cancelled by the opposite force contributed by ΔL. The parallel force has the following property: In the upper portion of the blade, it has a smaller magnitude and points in the negative y-direction. In the lower portion, it has a larger magnitude and points in the positive y-direction. Let fy_i be the y component of the parallel force. It can be computed by:

$$(7.4) \qquad fy_i = (C_1 - C_2y_i)\, e^{2\alpha_i} \sin(\alpha_i)\cos(\alpha_i)\,\Delta L$$

where $C_1=Ae^B[\gamma(H+y_0)\tan(\phi)+ c]$ and $C_2 = Ae^B\gamma\tan(\phi)$.

Now we calculate the summation of all fy_i's, represented by F^y, which gives us the total force pushing the soil mass in front of the blade upwards.

$$(7.5) \qquad F^y = \frac{1}{2}\sum_{i=1}^{n} (C_1 - C_2y_i)\, e^{2\alpha_i} \sin(2\alpha_i)\,\Delta L$$

To get an accurate solution, we let $\Delta\alpha$ approach 0. In this case we have the following equations [9]:

$$(7.6) \qquad \alpha_i = \alpha_0 + i\,\Delta\alpha$$

$$(7.7) \qquad \lim_{\Delta\alpha\to 0} \Delta L = R\,\Delta\alpha$$

$$(7.8) \qquad \lim_{\Delta\alpha\to 0} y_i = y_c - R\cos(\alpha_0 + i\,\Delta\alpha)$$

Replacing α_i, ΔL and y_i in (7.5) by right hand sides of above equations and making $\Delta\alpha$ infinitesimal, we obtain:

$$(7.9) \qquad F^y = \frac{R}{2}\int_{\alpha_0}^{\alpha_n} [C_1 - C_2y_c + C_2R\cos(\alpha)]\, e^{2\alpha}\sin(2\alpha)\, d\alpha$$

To simulate cases in which the blade are not fully loaded, we fix the lower bound angle of the definite integral and keep the upper bound angle changing from α_0 to α_n. That will give us the following figure:

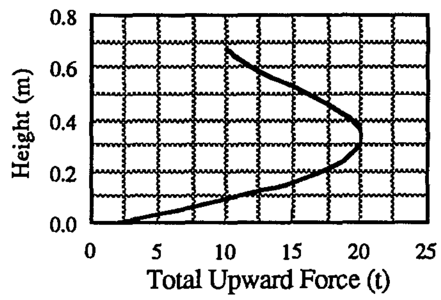


Fig. 9: Total upward force along the blade

In Fig 9, the vertical axis indicates y coordinates of points up to which the soil is loaded and the horizontal axis gives F^y under the given configuration. The data is recorded with α_0=1.22, R=100cm, c=1.9, δ=0.5, ϕ=0.54 and γ=2.0 (angles are measured in radius). For example, if the soil is loaded up to the middle point of the height of the blade, i.e. y=36.0 cm, the curve shows that at this point the total upward force reaches its maximum (about 20 metric tons).

The analysis shows that the total force is always positive. That is, the soil mass being cut always moves upward along the blade. This phenomenon is also observed experimentally [1]. The sequence of events occurring during the process of interaction between the cutting blade and the excavated soil before the blade can be described by 3 steps. 1) the soil chip being cut from the main soil mass moves upward along the blade because of resistance to the soil. 2) the soil chip is broken up into individual lumps on the upper part of the blade. 3) These lumps move downward toward the soil layers being further cut and from the soil prism which is being dragged. This phenomenon is depicted by Fig. 10:

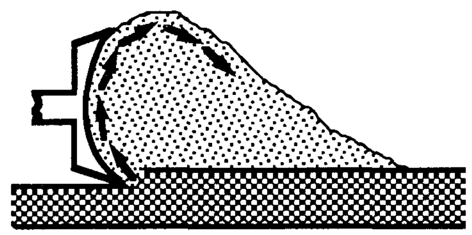


Fig. 10: Pattern of soil movement ahead of the blade

8. SOIL IN A BUCKET

In this section, we present a graphical model of a scooploader. For clarity, we assume that only buckets which can be represented by convex polygons will be processed. Again we first divide the soil configuration and the bucket into n slices with equal width Δx. This is shown in Fig 11, where the thick line segments indicate the bucket:

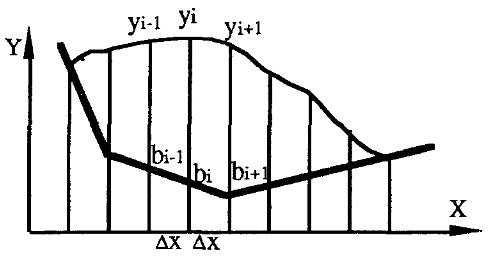


Fig. 11: Dividing the soil mass in a bucket

The motion of the soil mass in the bucket is a combination of two movements: 1) the movement of a portion of the given soil mass along a potential failure plane on the top; and 2) the whole mass along the bucket surface. We will refer to these motions as *local movement* and *global movement* respectively. In general, a local movement is caused by an unstable configuration of the given soil, while a global movement is due to the shear stress experienced by a surface of the soil mass in contact with the bucket. This can be seen more clearly through the free body diagram of slice-i arbitrarily picked from the partitioning (see Fig. 12), where f_i is the force driving a local movement along the failure plane denoted by line segment $<y_{i-1}, h_i>$. This force can be quantified by (3.7).

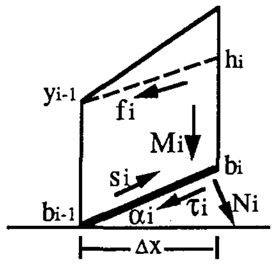


Fig. 12: Analyzing forces of slice-i

Let's now consider the global movement. The driving force, denoted by G, can be calculated by analyzing the free body diagram of each free body. As shown in Fig. 12, the weight M_i of slice-i can be decomposed into two elements: the shear stress force τ_i and the normal stress force N_i. N_i is canceled by the opposite force provided by the bucket surface $<b_{i-1}, b_i>$. τ_i is the force which pushes the mass to move along the bucket surface. The shear strength force s_i, on the other hand, resists the shear displacement of soil particles along the bucket surface. These forces can be determined from the Mohr-Coulomb theory as indicated by [5]:

(8.1) $\tau_i = M_i \sin(\alpha_i)$

(8.2) $s_i = cL_i + M_i \cos(\alpha_i) \tan(\delta)$

where c is the coefficient of cohesion, δ is the angle of external friction, L_i is the length of the line segment from b_{i-1} to b_i, M_i is the weight of slice-i, and α_i is the angle between the bucket surface and the horizontal. δ indicates a measure of the friction between soil and the surface of the bucket. It is given in radians. For loamy clay and sand, the typical values of δ are 18 and 30 respectively [1]. The units of these symbols are as explained earlier.

For equilibrium consideration, we use a method similar to the one described in [6]. The stress force τ and the strength force s can be expressed by vectorial summations:

(8.3) $\tau = \sum_{i=1}^{n} \tau_i = \sum_{i=1}^{n} M_i \sin(\alpha_i) <\cos(\alpha_i), \sin(\alpha_i)>$

(8.4) $s = \sum_{i=1}^{n} s_i = \sum_{i=1}^{n} M_i \cos(\alpha_i) \tan(\delta)<\cos(\alpha_i), \sin(\alpha_i)>$

Note that the term cL_i is dropped from(8.4), since the cohesion coefficient c describes molecular attraction among like particles and is zero between soil and a bucket surface. Thus, the safety factor F_s can be defined as

(8.5) $F_s = \dfrac{|s|}{|\tau|}$

When F_s is less than one, sliding of the whole mass along the bucket surface is inevitable. In this case, the total driving force G of the global movement can be computed by

(8.6) $G = \begin{cases} \tau - s, & \text{if } \tau > 0 \\ \tau + s, & \text{otherwise} \end{cases}$

In order to simulate the movement of soil mass in a bucket, we decompose G to smaller components which are parallel to the bucket surface. These component forces are distributed to slices so that the dynamics of soil can be considered individually for each slice. After carefully analyzing the behavior of the soil mass, we know that the following constraints must be satisfied:

1) The summation of component forces should equal G;
2) All slices should have the same x-acceleration.

The first constraint is obvious. The second one should be always true simply because: 1) a bucket always has a convex shape; and 2) some slices would fall apart and tension cracks or deformation would occur if the x components of their accelerations are different.

Let $G^x = G\cos(\alpha)$ and $G^y = G\sin(\alpha)$ be the x and y components of G respectively, where α is the angle between the vector G and the horizontal. Let g_i be a component force of G, which is experienced by the bucket surface of i-th slice. From the constraints we have

(8.7) $G\cos(\alpha) = \sum_{i=1}^{n} g_i \cos(\alpha_i)$

(8.8) $G\sin(\alpha) = \sum_{i=1}^{n} g_i \sin(\alpha_i)$

(8.9) $\dfrac{g_1}{M_1}\cos(\alpha_1) = \dfrac{g_2}{M_2}\cos(\alpha_2) = \cdots = \dfrac{g_n}{M_n}\cos(\alpha_n)$

(8.7)~(8.9) give us n+1 equations with n+1 unknowns, namely $g_1, g_2, \ldots g_n$ and α. Other variables can be computed according to the geometry of the given configuration. Solving the equations we obtain

$$(8.10) \quad g_i = \frac{M_i \cos(\alpha)}{M \cos(\alpha_i)} G, \qquad \text{for } i = 1, 2, \ldots, n.$$

where $\alpha = \tan^{-1}\left(\frac{1}{M}\sum_{i=1}^{n} M_i \tan(\alpha_i)\right)$.

Having f_i and g_i computed, we model the soil dynamics in a bucket by using Algorithm 1 to evaluate simultaneous equations in (5.3). In order to do so, we simply replace f_i by f_i+g_i when calculating the rate of changes of each post at the time t_{m+1}. The rest of the algorithm remains unchanged.

9. IMPLEMENTATIONS

9.1 IMPLEMENTATION OF A BULLDOZER

Recall that the terrain surface is represented by a regular tessellation model. An array, namely z, of size m×n is used to store the height of elevation posts. An element z(i,j) in the array represents the elevation at the location <i,j>.

As mentioned in section 7, an excavating process of a bulldozer can be separated into three phases. These actions can be simulated by an algorithm with three corresponding stages: digging, piling and soil slipping. First, the algorithm keeps track of the motion of the blade. If the altitude value of the bottom of the blade is denoted by b(i,j) at the location <i,j>, then any elevation post z(i,j) passed through by b(i,j) are forced to have the same value. This procedure will create a ditch along the path of the bulldozer on the terrain surface.

The second stage models the upward movement of the soil along the blade. Let P be a set of soil prisms which have been passed through by the blade in the last time step. Let $z_p(i,j)$, $z_q(i,j)$ and $z_r(i,j)$ be surrounding posts of a prism p(i,j). The amount of soil contributed by prisms in P to the soil chip moving upwards can be computed by:

$$(9.1) \quad V = \frac{\Delta x \Delta y}{6} \sum_{p(i,j) \in P} \Delta V(i,j)$$

where

$$\Delta V(i,j) = z_p(i,j)+z_q(i,j)+z_r(i,j) - b_p(i,j)-b_q(i,j)-b_r(i,j)$$

Finally, in the third stage the amount of soil computed by (9.1) is put in front of the blade. Since the height that the soil is lifted upward along the blade and the speed in which the soil chips are broken into individual lumps depend on the cohesion property of the given soil, the procedure can be simulated by spreading the soil to a chunk shown below:

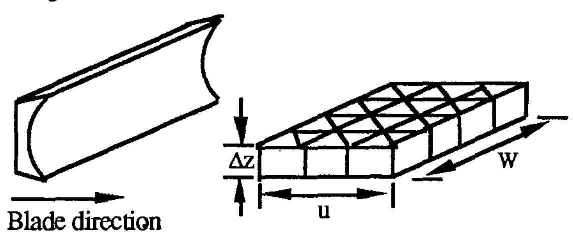


Fig. 13: Dimensions of soil chunk

The dimensions of the soil chunk are determined according to the following equation:

$$(9.2) \quad \Delta z = \kappa(1+c)$$
$$u = \frac{V}{w\,\kappa\,(1+c)}$$

where V is the total volume calculated by (9.1), w the width of the blade, c the cohesion coefficient, and κ a constant which determines how far forward the soil chip moves during one time step. In the implementation, κ is chosen experimentally to make the simulation looks more realistic.

After all this is done, Δz is added to the elevations of corresponding posts, and the slippage model introduced in previous sections is used to simulate the free flow motion of broken lumps of soil. It should be mentioned that the soil being brought to the top of the berm arrives continuously in the real world. However, with a discrete time simulation process, the chunk is a reasonable representation of the amount and location of the soil that would really arrive during one time step. The slippage model smoothly integrates this chunk into the berm, resulting in a realistic appearance.

Another important phenomena associated with physical properties of soil is swelling, which is due to a number of reasons: 1) the affinity of the soil for water; 2) the base exchange behavior and electrical repulsion; and 3) the expansion of entrapped air within the soil mass [4]. The model simulating the expansion of excavated soil is also discussed and implemented in [9].

9.2 IMPLEMENTATION OF A BUCKET

In implementing a 3-d bucket, we first divide it into m×n cross sections in a way such that they are parallel to either the x-z plane or the y-z plane. We refer to these sections as x-sections and y-sections respectively. The result of the division is shown in Fig. 14, where an x-section and a y-section are emphasized by two shaded polygons.

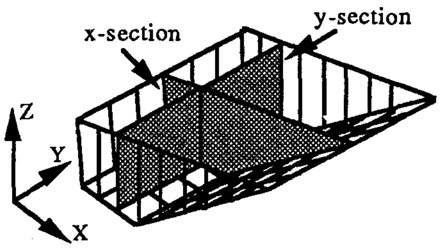


Fig. 14: Dividing a bucket into sections

Therefore, the 3-d soil dynamics in a bucket is reduced to m×n 2-d cases. For each individual cross section, we further partition a 2-d soil configuration into soil slices (see Fig. 11.). The soil dynamics of each slice is handled by means of the technique introduced in section 8.

A simulation procedure can be described as follows: Each iteration of a simulation can be accomplished by two steps. The first step computes forces for each soil slice of every bucket section according to (5.3) and (8.10). The second step uses the Euler integration method to determine new values for each elevation post (see Algorithm 1). These posts are intersections of x-sections and y-sections.

The cutting and loading activities of a scooploader can be modeled by a method similar to the one presented in section 7. The discussion, therefore, is omitted.

10. CONCLUSION AND FUTURE WORK

Experimental realtime models of a bulldozer and a scooploader have been implemented in the c programming language. Both time and space costs of algorithms are linear in the size of the bulldozer's blade, the size of the bucket and the resolution of the ground mesh. The simulations were done on a Silicon Graphics 4D/240 GTX computer. When using 4 processors, two bulldozers run at 6-8 frames/second. The scooploader model uses 2 processors, running at 10-15 frames per second. The number of elevation posts to model the ground for both models is 90×90. The simulations look very realistic.

Future research work may include soil compressibility and moisture content. The compression of soil layers is due to deformation and relocation of soil particles and expulsion of air or water from the void spaces [6]. Fundamental principles for estimating settlements of soil under superimposed loading should be explored so it can be used to provide vehicle tracks or conduct trafficability studies. The moisture content of the soil affects its unit weight and cohesion and results in different behaviors. Those properties should be incorporated into analytical models to provide more realistic simulations.

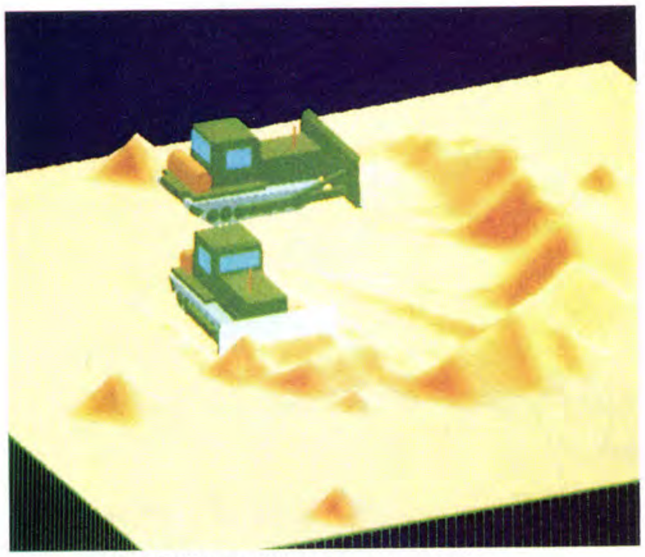


Fig. 15: Two bulldozers are at a work scene

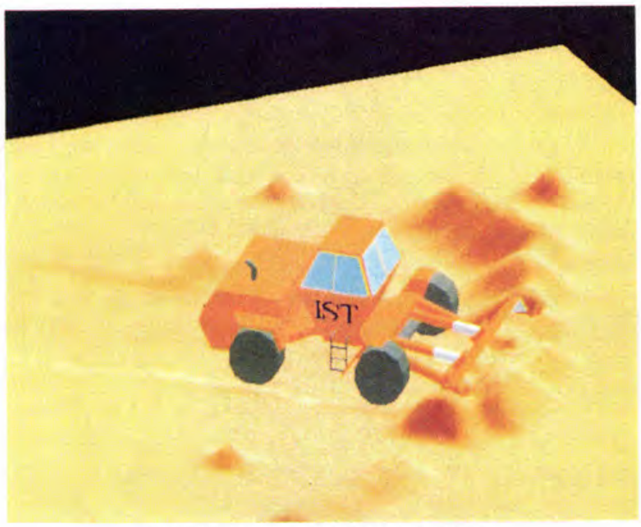


Fig. 16: A scooploader is loading

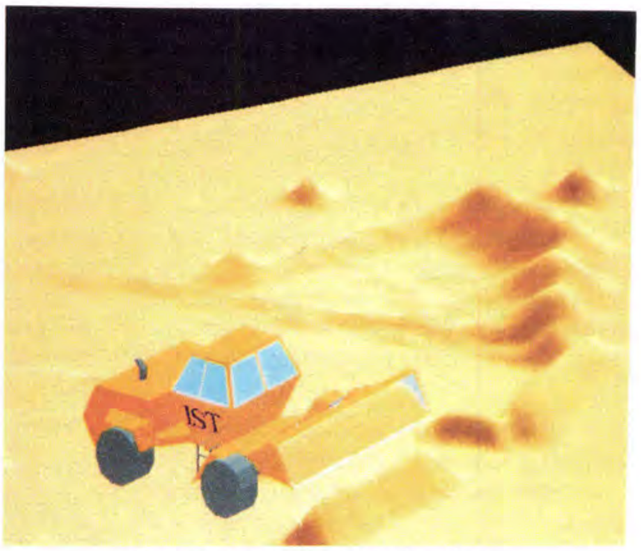


Fig. 17: A scooploader is dumping

ACKNOWLEDGMENTS

The authors are very grateful to Dr. Charles E. Hughes and Clay Johnson for revising the paper; to Lance Marrou and Jinxiong Chen for their support.

REFERENCES

1. Balovnev, V.I. *New Methods for Calculating Resistance to Cutting of Soil.* Translated from Russian, Published for the U.S. Department of Agriculture and the National Science Foundation. Washington, D.C., 1983.
2. Bromhead, E. N. *The Stability of Slopes.* Surrey University Press, 1986.
3. Burg, Jeniffer, Moshell, J. Michael, et al., Behavioural Representation in Virtual Reality. *Proceedings of Behavioral Representation Symposium.* Institute for Simulation and Training. Orlando, FL, 1991.
4. Cernica, J. N., *Geotechnical Engineering.* Holt, Rinehart & winston, 1982.
5. Chowdhury, R. N., *Slope Analysis.* Elsevier North-Holland Inc., 1978.
6. Das, Braja M. *Principles of Geotechnical Engineering.* Second Edition, PWS-KENT Publishing Company, 1990.
7. Huang, Y. H., *Stability Analysis of Earth Slopes.* Van Nostrand Reinhold Co., 1983.
8. Kass, Michael and Miller, Gavin. Rapid, Stable Fluid Dynamics for Computer Graphics. Proceedings of SIGGRAPH '90 (Dallas, Texas, August 6-10, 1990). In *Computer Graphics* 24, 4(August 1992).
9. Li, Xin. Physically-Based Soil Models of Dynamic Terrain in Virtual Environments. *Technical Report.* CS-TR-92-26, University of Central Florida. Nov. 1992.
10. Moshell, J. Michael. Li, Xin. et al. Nap-of-Earth Flight and the Realtime Simulation of Dynamic Terrain. *Proceedings of International Society for Optical Engineering.* Apr. 1990.
11. Winston, P. H., *Artificial Intelligence.* Addison-Wesley, pp.75-78, 1984.

Turbulent Wind Fields for Gaseous Phenomena

Jos Stam
Eugene Fiume[0]

Department of Computer Science
University of Toronto
10 King's College Circle
Toronto, Canada, M5S 1A4

Abstract

The realistic depiction of smoke, steam, mist and water reacting to a turbulent field such as wind is an attractive and challenging problem. Its solution requires interlocking models for turbulent fields, gaseous flow, and realistic illumination. We present a model for turbulent wind flow having a deterministic component to specify large-scale behaviour, and a stochastic component to model turbulent small-scale behaviour. The small-scale component is generated using space-time Fourier synthesis. Turbulent wind fields can be superposed interactively to create subtle behaviour. An advection-diffusion model is used to animate particle-based gaseous phenomena embedded in a wind field, and we derive an efficient physically-based illumination model for rendering the system. Because the number of particles can be quite large, we present a clustering algorithm for efficient animation and rendering.

CR Categories and Subject Descriptors: I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism; I.3.3 [**Computer Graphics**]: Picture/Image Generation; G.3 [**Probability and Statistics**]: Probabilistic algorithms.
Additional keywords and phrases: turbulent flow, stochastic modelling, Kolmogorov energy spectrum and cascade, transport model of illumination, Fourier synthesis, advection-diffusion, gaseous phenomena.

1 Introduction

We have come to appreciate the central role that irregularity plays in modelling the shape of natural objects. The analogue for wind and fluids is *turbulence*, and its effects are no less essential to the realistic portrayal of gaseous natural phenomena: curling wisps of smoke, mist blowing across a field, car exhaust, an aerosol spray, steam rising from a coffee mug, clouds forming and moving across the sky, the fall of leaves, a swirl of dust in a room, a hurricane. These effects are caused by the interaction of objects with a wind velocity field. Modelling the effect of wind requires that we model both the wind field and this interaction. Both Sims [14] and Wejchert and Haumann [17] model a wind field as the superposition of deterministic fields. Modelling a visually convincing turbulent wind field this way is painstaking. The greatest success in this direction was the particle-based "Blowing in the Wind" animation by Reeves and Blau [10].

Stochastic modelling is a natural alternative strategy. In [13], Shinya and Fournier describe an approach developed independently of ours but which has some similarities. They employ stochastic processes and Fourier synthesis to derive a wind field in spatiotemporal frequency domain, and invert the result to get a periodic space-time wind field. We employ the same paradigm, but our model and application are quite different. Although both wind models can be applied to a wide range of phenomena, and [13] demonstrates this very well, their main concern is with coupling the wind model to macroscopic physical models of rigid or deformable objects, whereas we are mostly concerned with microscopic interaction with gaseous and fluid phenomena. Consequently, our model of turbulence is dissimilar: Shinya and Fournier assume a constant deterministic temporal evolution (Taylor Hypothesis), while for us temporal evolution is also a stochastic process. Our wind model also differs in that an animator has direct control over deterministic and stochastic components of a field.

In this paper, turbulent wind fields are modelled as stochastic processes. The model is empirically plausible[5]. A wind field is generated from large-scale motion and from the statistical characteristics of the small turbulent motion, both freely chosen by an animator. This is analogous to modelling rough terrain by providing the global shape as given by a set of height samples, and the desired roughness of the terrain [2]. The large scale of the wind field will be modelled using simple wind field primitives [14, 17]. The small scale of the wind field will be modelled as a three-dimensional random vector field varying over space and time. This field is generated using inverse an FFT method[16] that we have generalized to a vector field. The resulting wind field has two desirable properties. First, it is periodic and is thus defined for any point in space-time. Second, it is generated on a discrete lattice and can be interactively calculated using four-linear interpolation.

Gases have been modelled in several ways. Ebert models a gas as a solid texture. With some trial-and-error (and in our experience, significant human effort), realistic animations were obtained[1]. Sakas models a gas as a 3-D random density field, generating it using spectral synthe-

[0]The financial support of the Natural Sciences and Engineering Research Council of Canada and of the Information Technology Research Centre of Ontario is gratefully acknowledged. The helpful suggestions of the referees are greatly appreciated.

sis [12]. While spectral synthesis is useful in generating turbulent wind fields, it is not ideal for directly generating density fields: visual artifacts appear due to the periodicity of the field and the entire density field must be computed at once. The temporal evolution of the density field is limited to simple translations. Both of the above models are computationally expensive to visualize, and hence interactive modelling is not feasible. Using physically-based turbulence to animate density fields is mathematically nontrivial, but we shall show that this can be done efficiently.

We model gases as density distributions of particles. The evolution of a density distribution within our wind field is described by an advection-diffusion equation. We efficiently solve this equation by modelling the gas as a "fuzzy blobby" with time varying parameters. A fast ray-tracing algorithm is used, based on a front to back single-scattering illumination model, to render such a density distribution.

2 A Multiple-Scale Wind Field Model

Physically, wind fields are the result of the variations of the velocity $\mathbf{u}(\mathbf{x},t)$ and the pressure $p(\mathbf{x},t)$ of a fluid (including air) over space and time. These variations are caused by various forces: external forces $\mathbf{F}$ applied to the fluid, nonlinear interactions between different modes of the velocity field and viscous dissipation at a rate ν. By summing these forces and equating them to the acceleration of the fluid we obtain the Navier-Stokes equations:

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u}\cdot\nabla)\mathbf{u} - \frac{1}{\rho_f}\nabla p + \nu\nabla^2\mathbf{u} + \mathbf{F}, \qquad (1)$$

where ρ_f is the density of the fluid. If the velocities of the fluid are much smaller than the speed of sound, we can assume that the fluid is incompressible [5], i.e.,

$$\nabla\cdot\mathbf{u} = 0. \qquad (2)$$

When proper initial conditions and boundary conditions are specified, Eqs. 1 and 2 are sufficient to solve for the velocity field and the pressure of the fluid for any time instant.

The above equations could be used to animate realistic wind fields. One would first specify the physical properties of the fluid that make up the model, including an initial velocity field and boundary conditions. One would then control the fluid motion by applying external forces. Realistic wind fields would be obtained by solving the Navier-Stokes equations as needed. This is entirely akin to the control problem for articulated figures, and it shares the same difficulties. First, a desired effect is hard to achieve by "programming" it using only external forces. Second, the non-linearities present in the Navier-Stokes equations make them hard to solve numerically, especially in the presence of turbulence (low viscosity). Linearizing the equations can improve stability and efficiency, which has been done by Kass and Miller to model the surface of water [4]. This results in highly viscous fluids that do not exhibit turbulence.

We shall model a turbulent wind field by separating it into a large-scale component $\mathbf{u}_l$ and a small scale component $\mathbf{u}_s$. The large-scale term is composed of simple wind fields, resulting in very viscous fluids. The small-scale term is a random field. We shall make a useful but physically implausible assumption that the components are independent, that is, that large scales do not affect the small scales and vice-versa. Hence we will write

$$\mathbf{u}(\mathbf{x},t) = \mathbf{u}_l(\mathbf{x},t) + \mathbf{u}_s(\mathbf{x},t). \qquad (3)$$

This assumption permits the real-time simulation and independent control of both large-scale and small-scale effects. The results, as we shall see, are quite convincing. We shall further discuss this assumption in our conclusions.

3 Small Scale Modelling

3.1 Random Vector Fields

In this section we will denote the small scale component $\mathbf{u}_s$ simply by $\mathbf{u}$. It is defined as a random space-time vector field, a function that assigns a random velocity to each point $(\mathbf{x},t)$ in space-time [15]. We shall invoke the standard Gaussian assumption [7]: that the random vector field is entirely determined by its second-order moments. These moments are obtained by statistically averaging (denoted by $\langle\ \rangle$) components of the evolving random velocity field. We will assume that the mean values of each component $\mu_i(\mathbf{x},t) = \langle u_i(\mathbf{x},t)\rangle$ $(i=1,2,3)$ of $\mathbf{u}$ are constant and equal to zero. The *cross-correlation* between different components of the velocity field at two different points in space-time $(\mathbf{x},t)$ and $(\mathbf{x}',t')$ are given by the functions

$$\Gamma_{ij}(\mathbf{x},t;\mathbf{x}',t') = \frac{\langle u_i(\mathbf{x},t)u_j(\mathbf{x}',t')\rangle}{\langle \mathbf{u}^2\rangle}, \quad i,j=1,2,3. \qquad (4)$$

Where $\langle\mathbf{u}^2\rangle = \langle u_1^2 + u_2^2 + u_3^2\rangle$ denotes the variance of the velocity field and physically is equal to twice the kinetic energy of the field. We will assume that the velocity field is *homogeneous* in space and *stationary* in time, which means that the cross-correlation only depends on the difference $\mathbf{r} = \mathbf{x}' - \mathbf{x}$ between the two points and the difference $\tau = t' - t$ between the two times: $\Gamma_{ij}(\mathbf{x},t;\mathbf{x}',t') = \Gamma_{ij}(\mathbf{r},\tau)$.

Homogeneous velocity fields have a corresponding representation in spatial-frequency domain via a spatial Fourier transform. Intuitively this transformation can be thought of as a decomposition of the velocity field into "eddies" of different sizes: large eddies correspond to small spatial frequencies and conversely for small eddies. The stationarity of the velocity field allows it to be represented in frequency domain by a temporal Fourier transform. We will denote spatial frequencies by $\mathbf{k} = (k_1,k_2,k_3)$ and temporal frequencies by ω.[1] We represent the velocity field in frequency domain via the usual Fourier transform:

$$\hat{\mathbf{u}}(\mathbf{k},\omega) = \int\int \mathbf{u}(\mathbf{x},t)\exp(-i\mathbf{k}\cdot\mathbf{x} - i\omega t)\,d\mathbf{x}dt. \qquad (5)$$

Writing the transform in this manner facilitates its separation into spatial and temporal frequency components. The Fourier-domain equivalent of the cross-correlation functions are the cross-spectral density functions:

$$\Phi_{ij}(\mathbf{k},\omega) = \langle \hat{u}_i^*(\mathbf{k},\omega)\hat{u}_j(\mathbf{k},\omega)\rangle, \quad i,j=1,2,3, \qquad (6)$$

where the "*" denotes the complex conjugation. Conveniently for us, the cross-spectral density functions and the cross-correlation functions are Fourier-transform pairs [15].

Finally, we assume that the velocity field is spatially *isotropic*, meaning that the cross-correlation functions are invariant under rotations. Thus the cross-correlation functions only depend on the distance $r = \|\mathbf{r}\|$ between two points. Isotropy and incompressibility (Eq. 2) imply that the cross-spectral density functions are of the form [5]

$$\Phi_{ij}(\mathbf{k},\omega) = \frac{E(k,\omega)}{4\pi k^4}(k^2\delta_{ij} - k_ik_j), \quad i,j=1,2,3, \qquad (7)$$

[1] In the turbulence literature, the term *wave number* is often used instead of *spatial frequency*. We will use *spatial frequency*, which is more common in computer graphics, but we shall denote spatial frequencies by $\mathbf{k}$, reserving the letter ω for temporal frequencies.

where δ_{ij} is the *Kronecker delta*, k is the length of the spatial frequency $\mathbf{k}$ and E is a positive function called the *energy spectrum function.* Its physical interpretation is that it gives the contribution of all spatial frequencies of length k and frequency ω to the total kinetic energy of the velocity field:

$$\frac{1}{2}\langle \mathbf{u}^2 \rangle = \int_0^\infty \int_{-\infty}^\infty E(k,\omega)\, d\omega\, dk. \tag{8}$$

3.2 The Energy Spectrum Function

Eq. 7 states that the structure of a velocity field (via its cross-spectral density functions) is entirely determined by its energy spectrum function. In other words, an animator can control the qualities of turbulent motion by specifying the shape of the energy spectrum. This function can be arbitrary as long as the integral of Eq. 8 exists. In the turbulence literature one can find a wide variety of different energy spectra for various phenomena. These models are either determined from experimental data or obtained from simplifying assumptions about the fluid. The best-known example of the latter for turbulence that has reached a steady-state (i.e., $\int_{-\infty}^\infty E(k,\omega)\, d\omega \rightarrow E(k)$) is the *Kolmogorov energy spectrum* [5]:

$$E_K(k) = \begin{cases} 0 & \text{if } k < k_{\text{inertial}} \\ 1.5\, \epsilon^{3/2}\, k^{-5/2} & \text{otherwise} \end{cases} \tag{9}$$

This spectrum results from an *energy cascade,* where energy introduced at frequency k_{inertial} is propagated to higher frequencies at a constant rate ϵ. Instead of invoking Taylor's Hypothesis [13] we model the temporal frequency dependence of the energy spectrum function $E(k,\omega)$ by multiplying the Kolmogorov energy spectrum $E_K(k)$ by a temporal spread function $G_k(\omega)$ subject to:

$$\int_{-\infty}^\infty E(k,\omega)\, d\omega = E_K(k) \int_{-\infty}^\infty G_k(\omega)\, d\omega = E_K(k). \tag{10}$$

This guarantees conservation of kinetic energy (cf. Eq. 8). Furthermore, we want the small eddies to be less correlated in time than the large eddies. Spatially, this means that small eddies spin, ebb and flow more quickly than large eddies; this behaviour can be observed when watching a water stream or smoke rising from a cigarette. We can achieve this behaviour by setting G_k to a Gaussian with a standard deviation proportional to k:

$$G_k(\omega) = \frac{1}{\sqrt{2\pi}\, k\sigma} \exp\left(-\frac{\omega^2}{2k^2\sigma^2}\right). \tag{11}$$

Indeed, for large eddies (as $k \rightarrow 0$), G_k is a spike at the origin, corresponding to the spectral distribution of a highly-correlated signal; for small eddies (as $k \rightarrow \infty$) the spectral density becomes constant, denoting an uncorrelated signal.

3.3 Generating the Small Scale Component

We now describe an algorithm to generate a random velocity field having specified cross-spectral density functions Φ_{ij}. The algorithm is a generalization of Voss's inverse FFT method[16]. The idea is to filter an uncorrelated white noise velocity field in the Fourier domain, and then to take an inverse Fourier transform to obtain the desired random velocity field. The challenge is thus to find the right filter such that the resulting velocity field has the desired statistics.

We first compute the velocity field in the frequency domain for discrete spatial frequencies (i,j,k) and temporal frequencies l.[2] Let us assume that the discretization is uniform and that there are N samples per dimension. Then the discrete Fourier transform (DFT) of the velocity field $\hat{\mathbf{u}}_{i,j,k,l}$ is defined on a discrete lattice of size $3N^4$. To ensure that the resulting space-time velocity field is real valued, the elements of the DFT must satisfy the following symmetries: $\hat{\mathbf{u}}_{i,j,k,l} = \hat{\mathbf{u}}^*_{N-i,N-j,N-k,N-l}$, where the indices are taken modulo N, i.e., $N-0 = 0$[9]. In the special case when the indices on both sides of the equality are identical (e.g., $\hat{\mathbf{u}}_{N/2,0,N/2,N/2}$) we have to set the imaginary parts of $\hat{\mathbf{u}}_{i,j,k,l}$ to zero. The following algorithm generates a DFT with the required properties.

```
for $i, j, k, l$ in $\{0, \ldots, N/2\}$ do
    compute $\hat{\mathbf{u}}_{i,j,k,l}$, $\hat{\mathbf{u}}_{N-i,j,k,l}$, $\hat{\mathbf{u}}_{i,N-j,k,l}$, $\hat{\mathbf{u}}_{i,j,N-k,l}$,
       $\hat{\mathbf{u}}_{i,j,k,N-l}$, $\hat{\mathbf{u}}_{N-i,N-j,k,l}$, $\hat{\mathbf{u}}_{N-i,j,N-k,l}$, $\hat{\mathbf{u}}_{N-i,j,k,N-l}$
    $\hat{\mathbf{u}}_{N-i,N-j,N-k,N-l} = \hat{\mathbf{u}}^*_{i,j,k,l}$
    $\hat{\mathbf{u}}_{i,N-j,N-k,N-l} = \hat{\mathbf{u}}^*_{N-i,j,k,l}$
    $\hat{\mathbf{u}}_{N-i,j,N-k,N-l} = \hat{\mathbf{u}}^*_{i,N-j,k,l}$
    $\hat{\mathbf{u}}_{N-i,N-j,k,N-l} = \hat{\mathbf{u}}^*_{i,j,N-k,l}$
    $\hat{\mathbf{u}}_{N-i,N-j,N-k,l} = \hat{\mathbf{u}}^*_{i,j,k,N-l}$
    $\hat{\mathbf{u}}_{i,j,N-k,N-l} = \hat{\mathbf{u}}^*_{N-i,N-j,k,l}$
    $\hat{\mathbf{u}}_{i,N-j,k,N-l} = \hat{\mathbf{u}}^*_{N-i,j,N-k,l}$
    $\hat{\mathbf{u}}_{i,N-j,N-k,l} = \hat{\mathbf{u}}^*_{N-i,j,k,N-l}$
end for
for $i, j, k, l$ in $\{0, N/2\}$ do
    set imaginary parts of $\hat{\mathbf{u}}_{i,j,k,l}$ to zero
end for
```

To compute each element $\hat{\mathbf{u}}_{a,b,c,d}$ in the first loop, three independent complex random variables $X_m = r_m e^{2\pi i \theta_m}$ $(m = 1,2,3)$ are generated with normally distributed gaussian random amplitudes r_m and with uniformly distributed random phases θ_m. The components of that element are then calculated as

$$\begin{aligned}
(\hat{u}_1)_{a,b,c,d} &= \hat{h}_{11}((i,j,k),l)X_1, \\
(\hat{u}_2)_{a,b,c,d} &= \hat{h}_{21}((i,j,k),l)X_1 + \hat{h}_{22}((i,j,k),l)X_2, \\
(\hat{u}_3)_{a,b,c,d} &= \hat{h}_{31}((i,j,k),l)X_1 + \hat{h}_{32}((i,j,k),l)X_2 + \\
&\quad\ \hat{h}_{33}((i,j,k),l)X_3.
\end{aligned}$$

The functions $\hat{h}_{mn}$ are derived from the cross-spectral density functions as shown in Appendix A (Eq. 21). The velocity field is then obtained by taking three inverse DFT's:

$$\begin{aligned}
u_1 &= \text{invFFT4D}(\hat{u}_1) \\
u_2 &= \text{invFFT4D}(\hat{u}_2) \\
u_3 &= \text{invFFT4D}(\hat{u}_3).
\end{aligned}$$

The resulting velocity field is defined on a discrete lattice and is periodic in space and time. Thus even a small lattice defines a field everywhere in space-time. The spacing of this grid determines the smallest scale of the turbulence.

4 Animation of Gaseous Phenomena

Physically a gas is composed of many particles. We could therefore animate a gas by moving its particles about the wind field, but this would require a vast set of particles. We shall instead consider the *density* $\rho(\mathbf{x},t)$ of particles at space-time point $(\mathbf{x},t)$. Assuming that the particles have no effect on the wind field, the evolution of the density distribution is given by an *advection-diffusion* (A-D) equation

[2]The choice of i, j, k here as indices should not be confused with their different use above.

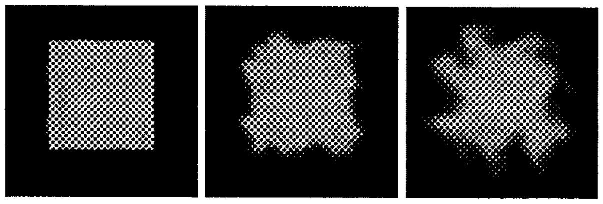


Figure 1: Evolution of a density distribution

[5] to which we have added a dissipation term:

$$\frac{\partial \rho}{\partial t} = -\mathbf{u}\nabla\rho + \kappa\nabla^2\rho - \alpha\rho. \qquad (12)$$

The first term on the right hand side is the advection term that accounts for the effects of the wind field on the density. The second term accounts for molecular diffusion at rate κ. This term can also be used to model turbulent diffusion from scales smaller than the smallest scale of the modelled turbulence. The third term accounts for dissipation of density at rate α. Since the velocity $\mathbf{u}$ is given, the equation is linear in ρ and can be solved by finite differences. The density distribution is then resolved on a finite grid and can be rendered using an efficient voxel-based volume renderer [1, 6]. Figure 1 depicts the evolution of an initially square distribution evolving under the influence of a two-dimensional wind field calculated using a standard PDE solver [9]. Computations for four-dimensional wind fields become rapidly prohibitive both in computation time and memory. To obtain tractable animations we propose an alternative strategy. We shall assume that the density distribution is a weighted sum of a simple distribution f:

$$\rho(\mathbf{x},t) = \sum_{i=1}^{n} m_i(t) f(\|\mathbf{x}-\mathbf{x}_i(t)\|, t-t_i) = \sum_{i=1}^{n} \rho_i(\mathbf{x},t). \quad (13)$$

In other words the density distribution is a "fuzzy blobby" with time-dependent field function f, where $\mathbf{x}_i(t)$ is the centre of mass, t_i is the time at which the "blob" ρ_i is created and $m_i(t)$ is its mass. If we suppose that f is a gaussian distribution with a standard deviation σ_0 much smaller than the smallest scale of the turbulent wind field, the wind field can be assumed to be constant on each blob. The advection term therefore only moves the blob, but does not deform its shape. The movement of the blob is hence given by integrating its centre of mass over the wind field:

$$\mathbf{x}_i(t) = \mathbf{x}_i(t_i) + \int_{t_i}^{t} \mathbf{u}(\mathbf{x}_i(s), s)\, ds, \qquad i = 1,\cdots,n. \quad (14)$$

The deformation of the shape of the blob is given by the diffusion term. Here we note that the diffusion at rate κ after time $t - t_i$ of a gaussian with variance σ_0^2 is equivalent to convolving a gaussian of variance $\kappa(t - t_i)$ with a gaussian of variance σ_0^2 (cf. [18]). Gaussians are closed under convolution, and the resulting gaussian has variance $\sigma_i^2(t) = \sigma_0^2 + \kappa(t - t_i)$:

$$f(r, t - t_i) = \frac{1}{(2\pi)^{\frac{3}{2}}\sigma_i^3(t)} \exp\left(-\frac{r^2}{2\sigma_i^2(t)}\right). \qquad (15)$$

Thus f diffuses outward with variance $\sigma_i^2(t)$ that increases with t. The normalization factor $(2\pi)^{\frac{3}{2}}\sigma_i^3(t)$ guarantees that the mass of the blob is invariant under diffusion. Once the variance of a blob becomes comparable to the smallest scale of the turbulent wind field we can replace it by smaller blobs and distribute the mass equally among them. The effect of the dissipation term is an exponential decay of the masses over time:

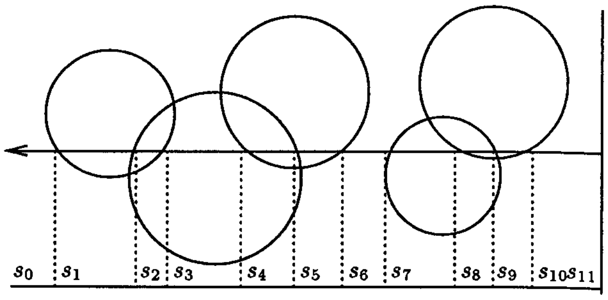


Figure 2: Subdivision of ray into intervals

$$m_i(t) = m_0 \exp\left(-\alpha(t - t_i)\right). \qquad (16)$$

5 Efficient Rendering of Gas

In conventional ray-tracing, light-object interactions are only computed at object boundaries. Hence light travelling along a ray is only modified at its endpoints. In the presence of a participating medium, the light carried by a ray can be attenuated and increased: attenuation is caused by light absorbed and scattered away by the gas; an increase may arise from light scattered in the direction of the ray from other directions and by self-emission of the gas. These effects can be included into a standard ray-tracer, by modifying the intensity value returned along any ray in the ray-tree. For each such ray we first determine which blobs have domains intersecting the ray (in practice we truncate the domain of each gaussian). For each such blob we store in a sorted list the parameter value s both for the entry and exit points of the ray. This subdivides the ray into N disjoint intervals $I_i = [s_i, s_{i+1}]$ $(i = 0,\cdots,N-1)$ as illustrated in Figure 2, with $s_0 = 0$ being the origin of the ray and the s_i being points of ray/blob intersections.

Once the ordered list of blobs intersecting the ray is calculated, the intensity of light C reaching the origin of the ray is computed by shading the list from front to back [6]:

```
$\tau_{total} = 1$
$C = 0$
for $i = 1$ to $N - 2$ do
    $C = C + \tau_{total}(1 - \tau_i)C_i$
    $\tau_{total} = \tau_{total}\tau_i$
end for
$C = C + \tau_{total}C_N$,
```

Here, τ_i is the transparency of the density distribution on interval I_i, and C_i is the intensity of light emitted on that interval by the density distribution. These values are defined in Appendix B, in which we also derive the illumination model. C_N is the intensity returned by the standard ray-tracer. In case the ray is cast to determine a shadow, only τ_{total} has to be returned.

The transparency along an interval I_i due to a single blob is a function only of the distance of the ray to the centre of the blob and the endpoints s_i and s_{i+1} of the interval as shown in Figure 3. The exact relationship and an efficient way to compute them is given in Appendix B. The transparency τ_i of the interval is then computed by

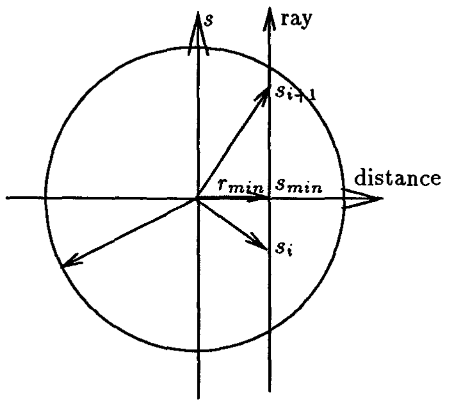


Figure 3: Calculation of transparencies τ_i

combining the transparency values calculated for each blob that intersects the ray along that interval.

Instead of testing separately for an intersection of the ray with each blob, we traverse a tree data structure of bounding spheres. The tree is constructed prior to rendering a frame as follows. First all the blobs are put in a linked list. The tree is then constructed by the following algorithm:

```
while list has at least two elements do
    for each blob $b$ in the list do
        search for blob $b'$ closest to $b$
        remove $b'$ from list
        create new blob $b''$ which bounds $b$ and $b'$
        set $b$ and $b'$ to children of $b''$
        replace $b$ by $b''$ in list
    end for
end while
```

There are some obvious optimizations that can be made to this brute-force algorithm, such as non-binary blob groupings and the use of a k-d tree to accelerate the search, but the cost of ray tracing overwhelms even brute-force preprocessing cost. On average, the use of the tree data structure has reduced rendering times by an order of magnitude. The tree can be thought of as a multi-scale representation of the density distribution and hence could be used to render the distribution at different levels of detail.

6 Interactive Field Modelling/Results

In our implementation, modelling wind fields and their effects consists of several steps. First the energy spectrum for the spatial component of the small-scale turbulence is specified by providing numerical values for the rate ϵ and the inertial frequency k_{inertial} of the Kolmogorov energy cascade. The standard deviation σ for the temporal component of the energy spectrum is also specified. The overall energy spectrum (cf. Section 3.2) is the product of the temporal and spatial (Kolmogorov) energy spectra. A 4-D vector field is then generated (cf. Section 3.3) which can be placed in a library (although its computation is swift).

We have developed an interactive animation system in which an animator can design a complex wind field and visualize its effect on a gas density. Complex wind fields are formed by the superposition of small-scale turbulence with large-scale fields such as directional, spherical, and exponentially decaying fields. The user is also able to change the grid spacing of the small scale independently in each component of space and time, allowing the specification of non-homogeneous fields. This also permits the same prototypical small-scale field to be given different behaviours in different contexts (which is precisely what has been for the images shown below).

Our animation system also simulates the effect of a wind field on a gas. A specific gaseous phenomenon is specified as a particle system characterized by the following values: the region over which blobs of particles are born, their birth rate, and the initial standard deviation and the initial mass of each blob. During a simulation, the system introduces blobs at the given rate, animates their motion by advection, modifies the standard deviations by diffusion and the masses by dissipation, as described in Section 4. Additionally, particles can be given illumination parameters such as a colour. In this modelling step the centre of each blob is depicted (with intensity modulated by parameters such as duration), but positions and other data can be piped into a high-quality renderer for image synthesis. About 6,000 particles can be animated in real time on an SGI Indigo.

The parameters needed for rendering include (Appendix B): the extinction coefficient κ_t, which describes the decay of light in inverse proportion to distance; the albedo $\Omega \in [0, 1]$, which defines the proportion of light scattered at a given point; the phase function p, giving the spherical distribution of scattered light; and self-emission Q, which is the amount of light emitted by a blob at a given position. The illumination computation for gas densities at a resolution of 640×480 typically requires from one to ten minutes, although 1-2 hour computations are possible when rendering scenes of high optical complexity.

For the images presented below, we have assumed that the phase function is constant and we have ignored shadows cast onto the density distribution for all but one image sequence. In all simulations the same statistical parameters were used for the small scale component: $\epsilon = 1$, $k_{\text{inertial}} = 4$ and $\sigma = 1$.

Steam from a mug: One global directional wind field was used to model the rising of the steam due to thermals. The particles were generated uniformly on a disk.

Psychedelic steam: Three trails of smoke of different colours were combined. As for the steam we used a directional wind field, this time tilted in the direction of the teapot spout. Particles were again generated on small disks.

Cigarette smoke: Two smoke trails originating from the tip of a cigarette are derived from the similar small-scale turbulence as the steam with a directional heat source.

Interaction of a sphere with smoke: This simulation shows how objects can interact with our wind field model. Instead of testing for collision of particles with the objects, we define a repulsion field around each object. We modelled the repulsion force by a radial potential field. The sphere is moved along a path given by a spline curve. Note that this image sequence depicts self-shadowing.

Three-dimensional morphing: The cylindrical range data of two human heads was converted into two sets of blobs and input to the animation system. The scene was illuminated by setting the self-illumination parameter (Q in Eq. 24) of each blob to the illumination given by the range data. The albedo was set to zero and dissipation was set to a large value to allow rapid dissolution of each set of blobs (with one run in reverse).

7 Conclusions and Extensions

We have presented a new model for the visual simulation of gaseous phenomena in turbulent wind fields. Our model provides an animator with control over both the large-scale motion and the statistical features of the small-scale turbulence. This model has been successfully applied to the animation of gaseous phenomena. Our model, however, can be applied to many other phenomena resulting from the interactions of objects with a wind field. For example, the wind field model can be included in any existing physically-based animation system. Our model can in fact generate a random vector field of any dimension, not only three-dimensional vector fields with a four dimensional domain. The derivation of the algorithm can be adapted in a straightforward manner. Our fast rendering algorithm can be used to visualize sparsely sampled data. The rendering of the heads in the morphing animation is a good example. Also our animation system could be used to visualize wind fields calculated by direct numerical simulation for fluid dynamics applications.

There are many other extensions to our model that we will explore in future research. We have assumed that the large scale motions of the wind do not modify the small turbulent scale. This is implausible. One possible solution is to warp the domain of the turbulent scale according to the large scales. We would require the use of a global deformation algorithm. Also it is possible to use a physical model for the large scales. A numerical technique in computational fluid dynamics known as *Large Eddie Simulation (LES)* solves the Navier-Stokes equations on a coarse grid using a statistical model for the small scales [11]. However, a physical simulation might not be relevant in computer graphics when a specific behaviour is intended.

A Inverse FFT Method Derivation

A white noise velocity field has cross-spectral density functions defined by:[3]

$$\Phi^w_{kl}(\mathbf{k},\omega) = \langle \hat{w}^*_k(\mathbf{k},\omega)\hat{w}_l(\mathbf{k},\omega)\rangle = \delta_{kl}. \qquad (17)$$

A random field with cross-spectral density functions Φ_{ij} can be obtained by *cross-convolving* this white noise with a set of deterministic kernels h_{kl}:

$$u_k(\mathbf{x},t) = \sum_{l=1}^{3}\int_{\mathbf{R}^3}\int_{-\infty}^{\infty} h_{kl}(\mathbf{x}-\mathbf{y},t-s)w_l(\mathbf{y},s)\,ds\,d\mathbf{y}, \quad (18)$$

which in the Fourier domain becomes

$$\hat{u}_k(\mathbf{k},\omega) = \sum_{l=1}^{3}\hat{h}_{kl}(\mathbf{k},\omega)\hat{w}_l(\mathbf{k},\omega). \qquad (19)$$

We obtain an equation for the transformed kernels $\hat{h}_{kl}$ in terms of the cross-spectral density functions Φ_{ij} by inserting the expressions for the Fourier velocity components $\hat{u}_i$ and $\hat{u}_j$ given by Eq. 19 into the definition of the cross-spectral density function Φ_{ij} (see Eq. 6).

$$\begin{aligned}\Phi_{ij}(\mathbf{k},\omega) &= \langle \hat{u}^*_i(\mathbf{k},\omega)\hat{u}_j(\mathbf{k},\omega)\rangle \\ &= \sum_{k=1}^{3}\sum_{l=1}^{3}\hat{h}^*_{ik}(\mathbf{k},\omega)\hat{h}_{jk}(\mathbf{k},\omega)\Phi^w_{kl}(\mathbf{k},\omega) \\ &= \sum_{n=1}^{3}\hat{h}^*_{in}(\mathbf{k},\omega)\hat{h}_{jn}(\mathbf{k},\omega). \qquad (20)\end{aligned}$$

We thus have 9 equations for the 9 kernels $\hat{h}_{kl}$ in terms of the cross-spectral density functions. Because of the symmetry of the cross-spectral density functions ($\Phi_{ij} = \Phi_{ji}$), only 6 of these kernels are independent and three kernels can be chosen arbitrarily. If we set $\hat{h}_{12} = \hat{h}_{13} = \hat{h}_{23} = 0$, then the system of equations given by Eq. 20 becomes diagonal and can easily be solved as follows.

$$\begin{aligned}\hat{h}_{11} &= \sqrt{\Phi_{11}}, \quad \hat{h}_{21} = \frac{\Phi_{21}}{\hat{h}_{11}}, \quad \hat{h}_{31} = \frac{\Phi_{31}}{\hat{h}_{11}} \\ \hat{h}_{22} &= \sqrt{\Phi_{22}-\hat{h}^2_{21}}, \quad \hat{h}_{32} = \frac{\Phi_{32}-\hat{h}_{31}\hat{h}_{21}}{\hat{h}_{22}} \\ \hat{h}_{33} &= \sqrt{\Phi_{33}-\hat{h}^2_{31}-\hat{h}^2_{32}}. \qquad (21)\end{aligned}$$

B Illumination Model

Consider a ray $\mathbf{x}_s = O + sD$, with origin O and direction D. Let C_N be the intensity of light reaching O along the ray from point $\mathbf{x}_b$ in the absence of a density distribution (i.e., given by a conventional ray-tracer). If we ignore multiple scattering effects, then the illumination C_0 reaching point O along the ray for each visible wavelength λ is [3]

$$C^\lambda_0 = \int_0^b \tau^\lambda(0,s)\rho(\mathbf{x}_s)\kappa^\lambda_t C^\lambda(\mathbf{x}_s)\,ds, \qquad (22)$$

where

$$\begin{aligned}\tau^\lambda(s',s'') &= \exp\left(-\kappa^\lambda_t\int_{s'}^{s''}\rho(\mathbf{x}_s)\,ds\right), \qquad (23)\\ C^\lambda(\mathbf{x}_s) &= \Omega^\lambda L^\lambda(\mathbf{x}_s) + (1-\Omega^\lambda)Q^\lambda(\mathbf{x}_s), \qquad (24)\end{aligned}$$

and κ_t is the *extinction coefficient*, and Ω is the *albedo*. The term $L(\mathbf{x}_s)$ is the contribution due to N_l light sources:

$$L^\lambda(\mathbf{x}_s) = \sum_{k=1}^{N_l} p^\lambda(cos\theta_k(\mathbf{x}_s))S_k(\mathbf{x}_s)L^\lambda_k, \qquad (25)$$

where p is the phase function characterizing the scattering properties of the density distribution, the θ_k are the angles between the ray and the vectors pointing to the light sources, S_k determines if the light source is in shadow and L_k is the colour of the light source. The term $Q^\lambda(\mathbf{x}_s)$ accounts for self-emission and can be used to approximate the effects of multiple scattering. If we assume that $C^\lambda(\mathbf{x}_s) = C^\lambda_i$ is constant on each interval I_i, which is reasonable in the case of many small blobs, then Eq. 22 becomes

$$\begin{aligned}C^\lambda_0 &= \sum_{i=0}^{N-1} C^\lambda_i \int_{s_i}^{s_{i+1}} \tau^\lambda(0,s)\rho(\mathbf{x}_s)\kappa^\lambda_t\,ds \\ &= \sum_{i=0}^{N-1} C^\lambda_i\left(\tau^\lambda(0,s_i)-\tau^\lambda(0,s_{i+1})\right). \qquad (26)\end{aligned}$$

If we define $\tau^\lambda_i = \tau^\lambda(s_i,s_{i+1})$ as the transparency along interval I_i then the equation becomes

$$C^\lambda_0 = \sum_{i=0}^{N-1}\left(\prod_{j=0}^{i-1}\tau^\lambda_j\right)C^\lambda_i\left(1-\tau^\lambda_i\right). \qquad (27)$$

We now show how the integral occurring in the calculations of the transparencies τ^λ_i can be computed efficiently. Let us assume that the blobs $\rho_{j_1},\cdots,\rho_{j_{n_i}}$ intersect the ray on interval I_i. The transparency on interval I_i is then

$$\tau^\lambda_i = \exp\left(-\kappa^\lambda_t\sum_{k=1}^{n_i}\int_{s_i}^{s_{i+1}}\rho_{j_k}(\mathbf{x}_s)\,ds\right). \qquad (28)$$

[3] All subscripted indices in this appendix take on the values 1, 2, 3.

As we render for a particular frame in time we define $\sigma_j^2 = \sigma_0^2 + \kappa(t - t_j)$ and $m_j = m_j(t)$. Using these definitions, each integral in Eq. 28 can be written as [8]:

$$\int_{s_i}^{s_{i+1}} \rho_j(\mathbf{x}_s)\,ds = \frac{m_j}{(2\pi)^{\frac{3}{2}}\sigma_j^3}\int_{s_i}^{s_{i+1}}\exp\left(-\frac{r_{min}^2+(s-s_{min})^2}{2\sigma_j^2}\right)ds$$

$$= \frac{m_j}{(2\pi)^{\frac{3}{2}}\sigma_j^2}\exp\left(-\frac{r_{min}^2}{2\sigma_j^2}\right)\left(T\left(\frac{s_{i+1}-s_{min}}{\sigma_j}\right)-T\left(\frac{s_i-s_{min}}{\sigma_j}\right)\right).$$

The first equality results from the geometry of Figure 3. The function T is the following integral:

$$T(s) = \int_0^s \exp\left(-\frac{u^2}{2}\right)du, \qquad (29)$$

and can be precomputed and stored in a table for efficiency.

References

[1] D. S. Ebert and R. E. Parent. "Rendering and Animation of Gaseous Phenomena by Combining Fast Volume and Scanline A-buffer Techniques". *ACM Computer Graphics (SIGGRAPH '90)*, 24(4):357–366, August 1990.

[2] A. F. Fournier, D. Fussell, and L. Carpenter. "Computer Rendering of Stochastic Models". *Communications of the ACM*, 25(6):371–384, June 1982.

[3] A. Ishimaru. *VOLUME 1. Wave Propagation and Scattering in Random Media. Single Scattering and Transport Theory.* Academic Press, New York, 1978.

[4] M. Kass and G. Miller. "Rapid, Stable Fluid Dynamics for Computer Graphics". *ACM Computer Graphics (SIGGRAPH '90)*, 24(4):49–57, August 1990.

[5] M. Lesieur. *Turbulence in Fluids: Stochastic and Numerical Modelling.* Kluwer Academic Publisher, Dordrecht, The Netherlands, 1990.

[6] M. Levoy. "Efficient Ray Tracing of Volume Data". *ACM Transactions on Computer Graphics*, 9(3):245–261, July 1990.

[7] J. P. Lewis. "Generalized Stochastic Subdivision". *ACM Transaction on Graphics*, 6(3):167–190, July 1987.

[8] N. Max, R. Crawfis, and D. Williams. "Visualizing Wind Velocities by Advecting Cloud Textures". In *Proceedings of Visualization '92*, pages 179–183, Los Alamitos CA, October 1992. IEEE CS Press.

[9] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C. The Art of Scientific Computing.* Cambridge University Press, Cambridge, 1988.

[10] W. T. Reeves and R. Blau. "Approximate and Probabilistic Algorithms for Shading and Rendering Structured Particle Systems". *ACM Computer Graphics (SIGGRAPH '85)*, 19(3):313–322, July 1985.

[11] R. S. Rogallo and P. Moin. "Numerical Simulation of Turbulent Flows". *Annual Review of Fluid Mechanics*, 16:99–137, 1984.

[12] G. Sakas. "Modeling and Animating Turbulent Gaseous Phenomena Using Spectral Synthesis". *The Visual Computer*, 9:200–212, 1993.

[13] M. Shinya and A. Fournier. "Stochastic Motion - Motion Under the Influence of Wind". In *Proceedings of Eurographics '92*, pages 119–128, September 1992.

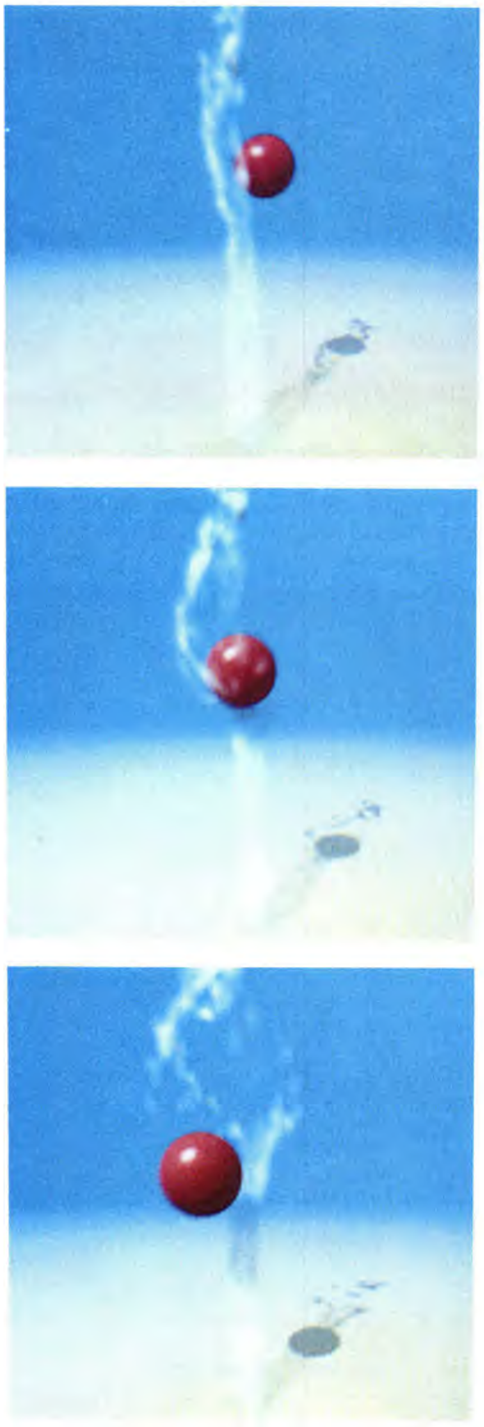
[14] K. Sims. "Particle Animation and Rendering Using Data Parallel Computation". *ACM Computer Graphics (SIGGRAPH '90)*, 24(4):405–413, August 1990.

[15] E. Vanmarcke. *Random Fields.* MIT Press, Cambridge, Massachussetts, 1983.

[16] R. P. Voss. "Fractal Forgeries". In R. A. Earnshaw, editor, *Fundamental Algorithms for Computer Graphics.* Springer-Verlag, 1985.

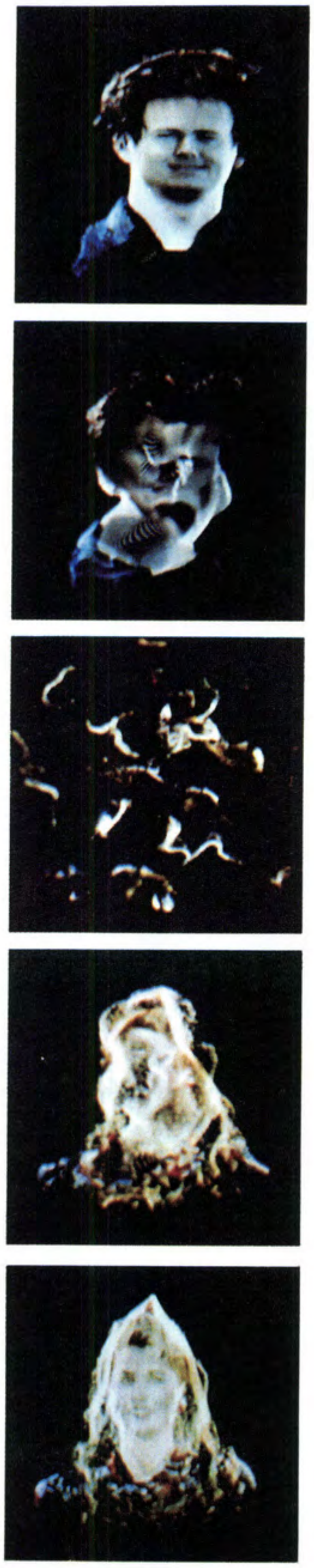
[17] J. Wejchert and D. Haumann. "Animation Aerodynamics". *ACM Computer Graphics (SIGGRAPH '91)*, 25(4):19–22, July 1991.

[18] A. Witkin and M. Kass. "Reaction-Diffusion Textures". *ACM Computer Graphics (SIGGRAPH '91)*, 25(4):299–308, July 1991.


Sphere interacting with a gas (note the shadowing)

A strange brew

The lonely cigarette



From David to Heidi

Real Virtuality: StereoLithography — Rapid Prototyping in 3-D

Chair:
Jack Bresenham, Winthrop University

Panelists:
Paul Jacobs, 3D Systems Inc.
Lewis Sadler, University of Illinois at Chicago
Peter Stucki, University of Zurich

Realistic Virtuality
Solid reality from virtual abstractions is now possible in mere minutes. Innovations in laser generation of 3-D objects offer rapid prototyping from computer synthesized graphics or scanned images to real solids in just a few hours or less. Photopolymers and thermoplastics offer new expectations for CAD, visualization, manufacturing, and medicine. Panelists from industry and academe will discuss current state-of-the art and expectations for the future of instant 3-D copies using new technologies such as StereoLithography, laser sintering, and fused deposition.

Panel Background
Real virtuality, in contrast to virtual reality, takes abstract images from computer synthesis and quickly turns them into actual 3-D objects as reality. This 'glimpse ahead' panel addresses use of StereoLithography, laser sintering and fused deposition as techniques for rapid prototyping. Engineering, manufacturing, medical, and artistic uses of this new technology offer significant growth potential as we enter the 21st century.

This panel brings together three leaders in innovative use and leading edge research for stereolithographic rapid prototyping. All of the panelists have been active in this new field of instant prototyping using laser-induced polymerization of photocurable resins. They will discuss industrial applications, biomedical usages, university research, associated software, and their views of what future challenges are likely in this rapidly developing technology.

Panel Goals and Issues
A glimpse ahead is the objective of this panel. Laser generated 3-D embodiments of virtual objects synthesized in CAD can be created as real physical objects in not much more time than was taken for 2-D computer plots a couple of decades ago. Panelists will discuss successful commercial applications and on-going research in which they're involved. They'll also address anticipated areas of activity such as data exchange standards, chemical & mechanical properties expectations, productivity enhancement and software paradigms.

The panel will first present views of 'real virtuality' or rapid prototyping together with illustrations of their work. To conclude the session they'll answer questions from attendees. If you've never seen laser generated 3-D parts nor held a computer fabricated polymer knee joint or gear box, the panelists will introduce you to this rapid prototyping in 3-D. If you're already quite knowledgeable and working in the area, the panel will share their research areas with you and expect you to comment and to question and to share your experiences of your own research and applications of StereoLithography.

Paul Jacobs
Currently over 370 StereoLithography Systems have been installed at major corporations, universities, government agencies and service bureaus in 25 countries on five continents. Dramatic cost and time savings have been achieved through the ability to rapidly proceed from the idea for an object to the object itself. The ability to hold a real object in your hands, to look at it from different directions, and to take advantage of the human brain's extraordinary pattern recognition capability has been one of the earliest benefits of this exciting new technology. We have referred to these capabilities for the early detection of design errors under the general headings of Visualization and Verification. The great majority of Rapid Prototyping and Manufacturing (RP&M) systems in current use have been justified on the basis of improved design visualization and verification.

However, during the past year we at 3D Systems have become aware of numerous applications of StereoLithography for design Iteration and Optimization. Substantial improvements in part accuracy, comparable to CNC machining, coupled with the proven ability to generate such objects very rapidly, at low cost and with greatly improved reliability, has now made it possible for designers and engineers to produce three, four or even five interactions within a few weeks. The result: improved product designs with fewer errors, available quickly, at lower cost.

Finally, we are now moving into an incredibly exciting era in which RP&M will enable the designer or engineer to achieve prototype and or limited run manufacturing Fabrication in final, end use, materials. With the release of the new QuickCast™ build style by 3D Systems, numerous users have already generated CAD models of a wide range of objects of varying complexity, produced a QuickCast quasi-hollow pattern of the object, and then, working directly with specific foundries, received precision shell investment castings of these objects in aluminum, stainless steel, beryllium cooper, titanium, silicon bronze, and incomel. The ability to generate functional prototypes, without the need for tooling, can save many months and tens to hundreds of thousands of dollars per component. Aggressively pursued, RP&M has the potential to significantly enhance industrial productivity.

Lewis Sadler
Rapid prototyping technology offers many advantages to biomedical applications that were previously unavailable. The ability to model complex, compound geometries is essential in the fabrication of maxillofacial prosthetics, facial implants, selected somato prostheses as well as anatomical models and simulators that have been developed to assist in surgical planning. Clinical experience has proven the efficacy of rapid prototyping technology as a new tool in the armamentarium of the surgeon in the restoration of facial cosmesis. The unusual educational mix of communication media (all of which are becoming digital) and the basic biological sciences (anatomy, pathology, physiology, histology, embryology, and genetics) and over twenty years of practical experience in communication problems for researchers and basic scientists have provided me with the necessary background to serve as liaison between groups of biomedical scientists and engineers. My experience is that these two

groups have no common background, no shared language or customs and act very much as separate nations, unable to communicate effectively with each other. BVL's success in this new technology area relates to the establishment of a multidisciplinary team in both the biomedical sciences and in engineering.

Peter Stucki

While a minimal set of algorithms and data structures have evolved for laser generation of 3D copies, further research and experimentation is necessary to achieve a good set of standard formats, algorithms, and tools. Abstract languages, formats for CT, PET and MRI scans, and specialized CAD/CAM software tools for rapid 3D object prototyping with StereoLithography, laser sintering and fused deposition are active research areas. Shrinkage compensation, cross-industry data exchange, and device characterizations are also fruitful areas for future advances. My multi-media laboratory at the University of Zurich has been involved for the past three years in the subject topic and I look forward to sharing our outlook. Of special interest are:

Chemistry for StereoLithography and Rapid Prototyping: The process of photopolymerization, e.g. the process of linking small molecules into larger molecules comprised of many monomer and oligomer units is key and will be shown as animated scientific visualization. For StereoLithography, radical and cationic processes of multifunctional monomers and oligomers resulting in cross linked polymers are of prime interest. Topics for discussion: building properties, mechanical characteristics.

Informatics for StereoLithography and Rapid Prototyping: 3D interactive computer graphics and natural image processing as well as their underlying algorithms, procedures and software tools represent the back-bone for StereoLithography and Rapid Prototyping object reconstruction. A systematic approach to classify processing options available includes the procedures of interactive and automated object design as well as procedures of interactive and automated object analysis. Data exchange standards such as the SLA and SLC formats are key in making StereoLithography and Rapid Prototyping applications platform independent. Yet, data volumes are enormous and very often represent the critical upper bound of what can be handled with ease in a given workstation environment. Topics for discussion: algorithmic efficiency, automatic and semi- automatic procedures, standards, hardware platforms, networks, application programming paradigms.

Post-Processing of StereoLithography and Rapid Prototyping Models: Topics for discussion: silicon casts, epoxy cast, metal cast, quick casting.

StereoLithography and Rapid Prototyping for CIM Applications: Topics for discussion: responding to fast changes, reducing time- to-market, total product modeling, prototyping and test quality control, pre-production marketing.

Jack Bresenham

In the late 1950's and early 1960's, research and use of numerical tool control was a hot 'computer' topic and leading edge graphics application. APT is a cooperative research & joint development effort I recall as having significant impact. Today laser generation of instant, or rapid prototype 3-D objects is a comparable new technology. It can allow blind persons to more easily 'view' mathematical functions, doctors to model individualized joints or restorations, and car manufacturers to model engine blocks without prohibitive cost and time delays that made typical tooling set-up for multiple models impractical. Our panelists each have well established reputations of long standing in other areas of computer graphics, imaging and visualization. For the past several years each has devoted significant research effort to advance the state of the art in laser generation of 3-D objects for rapid prototyping. I believe you'll find their insights into what I sometimes call instant 3-D hardcopy to be truly a glimpse ahead into what Dr. Stucki often calls real virtuality.

REFERENCES

Fuchs, W.A. and O. Trentz, StereoLithography for Medicine, International Workshop on StereoLithography in Medicine, April 1-2, 1993, University of Zurich, Zurich, Switzerland.

Hunziker, M., Chemistry for StereoLithography, International Workshop on StereoLithography in Medicine, April 1-2, 1993, University of Zurich, Zurich, Switzerland.

Jacobs, P., *Rapid Prototyping & Manufacturing: Fundamentals of StereoLithography*, SME, Dearborn, MI, 1992, (434 pages).

Marsh, J.L. and M.V. Vannier, Surface imaging from computerized tomographic scans, *Surgery*, 94, pp. 159-165, 1983.

Proceedings of the Second International Conference on Rapid Prototyping, University of Dayton, Dayton, Ohio, June 23-26, 1991.

Proceeding of the Third International Conference on Rapid Prototyping, University of Dayton, Dayton, Ohio, June 7-10, 1992.

Proceedings of the First European Conference on Rapid Prototyping, University of Nottingham, Nottingham, England, July 6-7, 1992.

Proceeding of the SME Rapid Prototyping & Manufacturing Conference, SME, Schaumberg, Illinois, September 29-30, 1992.

3D Hardkopien als Alternative zur 3D Visualisierung am Bildschrim, *Informatik Forschung and Entwicklung* 1992 (7), pp.121-125, Springer Verlag, 1992.

Visual Thinkers in an Age of Computer Visualization: Problems and Possibilities

Chair:
Kenneth R. O'Connell, University of Oregon, Eugene

Panelists:
Vincent Argiro, Vital Images
John Andrew Berton, Jr., Industrial Light & Magic
Craig Hickman, University of Oregon, Eugene
Thomas G. West, Author of *In the Mind's Eye*

We think that the same mind's eye that can justly survey and appraise and prescribe beforehand the values of a truly great picture in one all-embracing regard, in one flash of simultaneous and homogeneous comprehension, would also ... be able to pronounce with sureness upon any other high activity of the human intellect.
—Winston Churchill, *Painting As A Pastime*, 1932

It is now becoming increasingly clear that new technologies and techniques currently being developed in computer graphics, scientific visualization and medical imaging could have important implications in the larger society—in time having a profound effect on education and work at all levels. As visualization hardware and software become more sophisticated and are used more widely, there is a need to focus on differing abilities among individual users. Some are very good at processing visual material, while others find it an area of great difficulty.

For centuries, most of education and many occupations have been largely dominated by verbal approaches to knowledge and understanding. If current trends continue, it seems likely that there will be a gradual but dramatic reversal in many spheres, as powerful visualization techniques are used to find solutions to complex problems that are well beyond the limits of traditional modes of verbal and mathematical analysis. In many fields, visual approaches have been relatively unfashionable and under-used for about a century. Yet visual approaches have been major factors in the most creative and original work of a number of important historical physicists, chemists, mathematicians, inventors, engineers and others.

A dramatic revival of long-neglected visual approaches is already underway in several fields. Mathematicians are rediscovering the power and effectiveness of visual approaches that were long considered unacceptable. Calculus professors, with higher-powered graphic computers and well-designed software, are discovering ways to move rapidly to high level work, even with unexceptional students. Critics of engineering education in recent decades lament the excessive prestige of highly mathematical analysis and design while the "art" and "feel" and high visual content of previous design approaches have been denigrated—leading sometimes to major design failures. The revival of visual approaches is now increasingly apparent at the forefront of many fields. However, the high value of these approaches is nowhere more apparent than in the new fields that have been emerging in the last decade or so—chaos, fractals, system dynamics, complexity.

The continued spread of increasingly powerful and inexpensive graphic hardware and software (together with simulation and interactive media) can be expected to only further accelerate these trends—making it possible for many people to use methods and approaches that previously only a small number of extremely gifted people could apply through their own mental models alone.

Gradually, it is being recognized in some professions (such as engineering, medicine, architecture and scientific research) that those with high visual and spatial talents may have moderate or severe difficulties with certain verbal or numerical material—and that professional training programs that do not (formerly or informally) acknowledge this pattern may serve to eliminate some of the most talented (and ultimately most creative and productive) individuals.

Certain psychologists consider visual-spatial abilities a distinct form of intelligence, like logical or verbal intelligence, while certain neurologists suggest that there may sometimes be an inverse relationship between visual-spatial abilities and conventional verbal and academic abilities. Thus, there is a growing awareness that some very highly gifted visual thinkers may be expected to show a pattern of traits consistent with dyslexia or learning disabilities—having significant difficulties with reading, writing, composition, counting, speaking, memory or attention.

The late Harvard neurologist Norman Geschwind was interested in the apparently paradoxical pattern of high visual talents with verbal difficulties. He observed that "it has become increasingly clear in recent years that dyslexics themselves are frequently endowed with high talents in many areas. ... I need only point out the names of Thomas Edison and Albert Einstein to make it clear that dyslexics do not merely succeed in making a marginal adjustment in some instances, but that they rank high among those who have created the very fabric of our modern world." He suggested "that this is no accident." Rather, "there have been in recent years an increasing number of studies that have pointed out that many dyslexics have superior talents in certain areas of non-verbal skill, such as art, architecture, engineering, and athletics." He argued that the early formation of the brain may explain these patterns and should help us not to be surprised at those who have such mixed abilities.

It is possible, therefore, that conditions are being reversed in a way that will be especially favorable to some strong visual thinkers, many of whom may have had significant difficulties in conventional academic settings. With the further development of these technologies, we may see the development of a new visual language and striking new opportunities for creative, visual thinking persons. Increasingly, we may see them forming bridges between the arts, their traditional stronghold, and the scientific and technical fields that have been closed to many of them. We are used to hearing of scientists, computer programmers and mathematicians who are also talented musicians. Possibly in the future we may see the solution of complex problems in molecular biology, statistics, financial markets, neuroanatomy, materials development and higher mathematics coming from people who are sculptors, graphic artists, craftsmen, film makers and designers of computer graphic visualizations. Different kinds of tools and different kinds of problems may demand different talents and favor different kinds of brains.

As part of the panel, an "overview" presentation will be provided to describe relevant recent neurological and psychological research, to provide brief examples of historical and contemporary visual thinkers and to suggest the possible social, economic, educational and cultural implications of shifting from a predominantly verbal culture to one that is increasingly focused on visual approaches to learning, knowledge and experience. Subsequently, panelists will give presentations referring to their own work in computer graphics—generally representing perspectives related to scientific visualization, education and entertainment. The panelists will discuss and debate the extent to which they agree or disagree with the views put forth by other panelists and issues raised by the audience.

Vincent Argiro

In our need to comprehend space, volume visualization becomes an interdisciplinary adventure. Cells are 3D space-filling objects. Human brains are 3D space-filling objects. The bedrock under the Gulf of Mexico is an enormous 3D space-filling object. In each case we are curious, even desperate to know what lies inside these spaces. We want to see freely inside, with the mere intention to do so, like the Superman of our childhood with his X-ray vision.

We are a long way yet from fully actualizing this vision. But recent strides in digital imaging and computer graphics hardware and software suggest that this goal is becoming less remote. Moreover, in our own adventures with scientists, physicians and engineers, we find striking parallels in the specific visualization, analysis, modeling and communication tools these professionals require. To peer into and comprehend these regions of space, whether microscopic or macroscopic, living or inanimate, common approaches prevail over unique requirements.

This indeed suggests that digital imaging and volume visualization may be creating a fundamentally new visual language for communicating insights into the natural and artificial worlds. My presentation will make this case, illustrated with actual instances of overlap, osmosis and cross-pollination among investigations into cellular, neural, and geologic space.

John Andrew Berton, Jr.

Cinema is a medium where ideas are routinely communicated through primarily visual means. Films and videos are created by strong visual thinkers and viewers' interpretations of these works are based largely on visual information. This is especially true of visual effects work, where images must carry important content with little verbal assistance.

At Industrial Light and Magic, artists and technicians work with film directors and visual effects supervisors to bring important and demanding visual concepts to the screen. The techniques required to achieve these visual effects are often highly technical and, in many cases, based on logical systems, such as computers and computer graphics programs. The results of these techniques are judged in the visual realm, creating a unique opportunity to observe the translation of visual ideas into the verbal/technical realm and back again. Case studies of recent feature film projects at ILM indicate possible solutions to problems faced by visual thinkers in a verbal/technical arena. Topics of discussion include evaluations of how visual thinkers use existing images to describe their vision for images yet to be created and how ILM builds and uses interactive computer graphics tools to help visual thinkers communicate their ideas and create compelling visual effects.

Craig Hickman

While media attention centers on high-end scientific visualization and virtual reality as the epitome of what the computer can offer to visual thinking, a quieter but equally significant revolution is occurring on the desktop. Users are now expecting everyday software to display data in a visually satisfying way, and when they are confused using the software they are less apt to blame their confusion on their own ignorance rather they assume the fault lies with bad software design. The visual logic of software must be as well thought out as the logic of the data structures and software design teams will have greater need for "visual thinkers."

My own software "Kid Pix" attempts to provide a rich visual experience. It approaches this in two ways. First the user interface is as straightforward and as "self defining" as possible. Users are not expected to read a manual to get started and are invited to learn the program by exploration. Second, "Kid Pix" is full of randomness, visual surprises, and visual jokes. Even though these two approaches seem at odds, they support each other.

Thomas G. West

For some 400 or 500 years we have had our schools teaching basically the skills of a Medieval clerk—reading, writing, counting and memorizing texts. But with the deepening influence of computers of all kinds, it now seems that we might be on the verge of a really new era when we will be required to cultivate broadly a very different set of skills—the skills of a Renaissance thinker such as Leonardo da Vinci, recombining the arts and the sciences to create elegant and integrated solutions to urgent and complex problems.

As part of this change, in the not-too-distant future, past ideas of desirable traits could be transformed. In time, machines will be the best clerks. Consequently, in place of the qualities desired in a well-trained clerk, we might, instead, find preferable: a propensity to learn directly through experience (or simulated experience) rather than primarily from books and lectures; a facility with visual content and modes of analysis instead of mainly verbal (or symbolic or numerical) fluency; the more integrated perspective of the global generalist rather than the increasingly narrow specialist; a habit of innovation through making connections among many diverse fields; habit of continuous and life-long learning in many different areas of study (perhaps with occasional but transient specialization); an ability to rapidly progress through many stages of research and development and design using imagination and mental models along with 3D computer-aided design.

Leonardo da Vinci's emphasis on imitating nature and analysis through visualization may come to serve us as well as it served him—providing results well in advance of those who follow other more conventional approaches. Accordingly, in the near future, it seems that we might be in a position to come full circle, using the most sophisticated technologies and techniques to draw on the most elementary approaches and capacities—to simulate reality rather than describe it in words or numbers. To learn, once again, by doing rather than by reading. To learn, once again, by seeing and experimenting rather than by following memorized algorithms. Sometimes the oldest pathways can be the best guides into unfamiliar territory.

References

Brown, D., H. Porta and J.J. Uhl, "Calculus and Mathematica," in *The Laboratory Approach to Teaching Calculus*, L. Carl Leinbach et al, (eds.), Mathematical Association of America, 1991.

Ferguson, Eugene S., *Engineering and the Mind's Eye*, MIT Press, 1992.

Gardner, Howard, *Frames of Mind: The Theory of Multiple Intelligences*, Basic Books, 1983.

Geschwind, Norman, and Albert M. Galaburda, *Cerebral Laterialization: Biological Mechanisms, Associations and Pathology*, MIT Press, 1987.

Jolls, Kenneth R. and Daniel C. Coy, "The Art of Thermodynamics," *IRIS Universe*, No. 12, spring, 1990.

Jolls, Kenneth R.,"Understanding Thermodynamics through Interactive Computer Graphics," *Chemical Engineering Progress*, February 1989.

Kaufmann, William J., and Larry L. Smarr, *Supercomputing and the Transformation of Science*, Scientific American Library, 1993.

Ritchie-Calder, Peter R., *Leonardo & the Age of the Eye*, Simon and Schuster, 1970.

Satori, Giuseppe, and M. Mitchell Leonaop, *Complexity: The Emerging Science at the Edge of Order and Chaos*, Simon & Schuster, 1992.

West, Thomas G., "A Return to Visual Thinking," *Computer Graphics World*, November 1992.

West, Thomas G., *In the Mind's Eye: Visual Thinkers, Gifted People with Learning Difficulties, Computer Images and the Ironies of Creativity*, Prometheus Books, 1991.

Zimmerman, Walter, and Steve Cunningham (eds), *Visualization in Teaching and Learning Mathematics*, Mathematical Association of America, 1991.

Updating Computer Animation: An Interdisciplinary Approach

Chair:
Jane Veeder, San Francisco State University

Panelists:
Charlie Gunn, Technisches Universitat Berlin
Scott Liedtka, Forensic Technologies International
William Moritz, California Institute of the Arts
Tina Price, Walt Disney Pictures

Computer animation currently enjoys a wide range of applications from children's entertainment to disaster simulation, esoteric mathematics to personal fine art statements. In order to develop useful technology and train future professionals, we need to update our model of "Animation" and "Animator" into a pluralistic model that encompasses these and other applications. This panel will articulate and compare the conceptual framework, design gestalt, relation of design to content, and development process used by the very different animation application areas of entertainment, scientific and engineering visualization, and fine art as well as explore the connection between current forms and historical animation. Lively discussion will highlight points of commonality and contrast, reveal how these fields view each other and what each can learn from the other.

Panel Overview

With the zoetrope and other accessible contraptions, we began the transition from the era of the static media image to that of the dynamic. Motion film and video accelerated this transition and now digital technology is blurring the traditional differences between film and video, the optical and the synthetic image, science and art. For some, "Animation" is synonymous with "Cartoon." For others, it means "Dynamic Simulation.. Do we need new terminology? First, we need updated information: What is the mental model of animation and viewer? How will the needs of the application area drive the development of hardware and software? How does the use of computers impact the design and production process? What are the design constraints? How are design and production roles defined? What backgrounds do these animators have and what new skills must they learn? Who are valuable collaborators? How are "Time," "Space," and "Story" dealt with in each of these application areas? How do these animations relate to historical animation and other forms of contemporary representation? How can technical and media arts education better prepare people to work in these areas?

As animation joins writing as a basic skill for the communication of complex information, we must articulate the combined knowledge and experience of traditional and emerging practitioners. As computer animation, much like computer graphics, becomes a ubiquitous, enhancing technology, we must inform the development of that technology with our design experience and diverse creative goals.

Charlie Gunn

Mathematical animation represents a "return of the repressed": in its best moments, it reaches behind the symbolic artifacts preserved in textbooks to reveal the ding-an-sich of mathematical activity. In this presentation I'll make a case study of the mathematical animation, "Not Knot," tracing the decisions made to convert "equations" into "pixels." I suggest that the conversion is closer to archeology than discovery: behind the equations stands the original human imagination which in many ways is closer to a picture than to an equation. Part of the historical mission of scientific visualization is to reclaim the original form of much scientific creativity, which is intuitive, plastic, dynamic.

Animation is a key ingredient in this revelation by showing the mathematical universe to be alive with metamorphosis and movement. Attempting to open a window onto this universe using standard animation systems is often frustrating. I'll examine how the modeling, animation, and rendering requirements of "Not Knot" could not be met by existing monolithic systems. How can this situation be improved? I propose some guidelines for the design of open and flexible animation systems that have "room to grow" as new directions of mathematical exploration are pursued. Results obtained would have application to wider realms of animation practice. As an example, there are several places in "Not Knot" where transitions to infinity occur. Through integrating such limiting processes into animation technique we can begin to understand and represent qualitative metamorphosis, a key feature of much interesting animation. Such a system would also improve the aesthetic component of mathematical animation by simplifying collaboration with artist/ designers. Going in the other direction, it would also be of interest to artists working in the non-representational animation tradition of Fischinger, Whitney, and Cuba.

Scott Liedtka

At FTI, we produce for the courtroom computer animations that recreate accidents or explain technical processes. In either case, we attempt to teach technical concepts to lay people who must use them to make serious decisions. In doing so, we design our animations to draw attention to important events and to the relationship between different events. We often study an event by changing its timing, speeding it up or slowing it down, using repetition, or even reversing it. Also, our animations are often shown out of order as well because presenters using a bar coded laserdisc have random access to the animation segments we produce. Forensic animation has a special and legal relationship to reality. For instance, most of our motion is based upon recorded real world data, witness testimony, or established dynamics formulas. Camera views may be that of an eye witness or just an unglamorous close-up of a part in the process of failing. Although the technical aspects of our work are significant, it is also important to keep our audience's attention. Traditional animation techniques and timing as well as good design are crucial to keeping our audience watching and learning but we must not cross over into the kind of graphical fictions so successful in the entertainment domain. Our staff, a combination of mechanical engineers with computer artists and media specialists, reflects this balancing act.

Tina Price

In an era of dramatic change and expansion in animation, it is useful to try to define terms and take a new look at convention. I want to look at the long-used design philosophies of full character animation and discuss the difference between character animation and just moving things around. Using the Magic Carpet in "Aladdin" as an example, I'll show and discuss why character animation is so tied to human input and why character animation techniques are effective whether you're using a pencil, a potato or a computer. Just as the application of computers to character animation is changing the complexion of our development process so is "Computer Animation" being transformed by the application of character animation. In order to articulate our current situation, it is useful to identify some aesthetic parallels between the growth and development of early, hand-drawn animation and early computer animation.

Contemporary character animation filmmaking integrates images generated by a variety of techniques and technologies including hand drawn work. Using examples from Walt Disney's Feature Animation Department of computer animation elements from 1986-1993 I will discuss how computers have fit into our traditional animation work flow and, in turn, how they have changed how we view, develop, and produce animation. Some animation work roles have changed dramatically with the introduction of computers, with several traditionally separate roles collapsed into the single role of "Computer Animator."

Computer animation products have recently begun to actively integrate character animation principles and techniques. What direction can animation hardware and software take to support character animation more fully? And how can character animation training change to adjust to the expanding role of computers?

Jane Veeder

Education Fine Arts Animation: Like many others, I moved without benefit or burden of traditional animation training into computer animation as an extension of analog dynamic media (e.g. video, video synthesis) as small digital computers with graphics capabilities appeared in the late 1970's, offshoots of the booming videogame industry. As a fine artist, I have a central interest in digital technology as an inherently dynamic art medium [italics], not merely one of many animation "tools." In addition, much art theory surrounding the fine artist derives from the impact of computers and telecommunications on human culture. Thus I would work with other computer-based media before I would work with non-electronic animation techniques. In the same vein, I appropriate into my work formal conventions and modes of representation from other areas of computer graphics/animation. My design and development process is an interactive, evolutionary one whose open endedness reflects the nature of the medium. Through this method I decided what to do in large part, no simply to do what I have decided. This self-consciousness about the process of interacting with the medium and incorporating that, formally and conceptually, into the artwork is a habit of fine artists. This motivates us to try to interpret the medium to contemporary culture rather than retell old stories in a cost effective or more visually compelling manner.

Animation Education: In all areas of contemporary life boundaries between disciplines are eroding, driven primarily by technological opportunity. Common digital workspaces are arising between the traditional arts disciplines and between the arts and sciences. Fine arts or independent animation may seem a tiny island in a sea of commercial production and engineering/scientific visualization but it is here that most future computer animators are being trained. Soon, animation education everywhere will entail training in both physical and digital media and a merging of vocabularies, techniques, and wisdom. Even more computer and physical science majors will take our courses and more art students will get computer science degree minors. Inspired by the example of a few, early interdisciplinary graduate programs and incorporating new opportunities such as multimedia and virtual reality, we can articulate a new curriculum that embraces a wide range of creative and vocational potential and encourages students to prototype interdisciplinary collaborations.

William Moritz

Since computers are merely tools, "Computer Animation" is not separate from animation produced by other means, so it must be evaluated comparatively. The pin-screen of Claire Parker and Alexander Alexeieff contained 500,000 articulate light-points — quite like pixels — and their 1933 "Night on Bald Mountain" uses them for astonishing transformations of human forms; similarly PDI's morphing for Michael Jackson's "Black or White" video makes brilliant use of transformations to reinforce a social message, while the banal use of morphing in ads for soda pop and autos seems futile. John Lasseter's "Luxo, Jr." is an excellent character animation, even using the limitations of his programs (the rather glossy plastic/metallic surfaces) as part of the characters. Larry Cuba's "TWO SPACE" is an outstanding abstract animation, with a conceptual framework of positive and negative space that gives it a metaphysical resonance: inclusion/exclusion, matter/anti-matter, being and nothingness and creation out of nothingness. "Not Knot" is fine scientific educational animation in that it teaches not only spatial geometry but also our perceptual experience of it and how we learn about our world through seeing.

Summary

Animation is a field whose knowledge base and diversity of application is expanding rapidly. In order to develop useful tools and train successful students, we need to know more about how various application areas are using animation, how animation is impacting the applications, and what new problems, concepts, and opportunities for creativity are emerging. With this panel, we do not hope to close the discussion with definitive answers to all attendant questions, but rather to open it wide.

Recommended Reading

John Berger, *Ways of Seeing*, Penguin Books, 1988
John Canemaker, Ed., *Storytelling in Animation*, American Film Institute, 1988
Andrew Glassner, *3D Computer Graphics: A Guide for Artists and Designers*, Design Press, 1989
Donald Graham, *Composing Pictures, Still and Moving*, Van Nostrand Reinhold, 1983
Roger Noake, *Animation*, MacDonald, 1988
Frank Thomas and Ollie Johnston, *Too Funny for Words*, Abbeville Press, 1987
Edward R. Tufte, *Envisioning Information*, Graphics Press, 1990
Robert Russett and Cecille Starr, *Experimental Cinema*, DeCapo Press, 1988
Jack Soloman, *The Signs of Our Times*, Harper & Row, 1988
Brian Wallis, Ed., *Art After Modernism: Rethinking Representation*, Godine, 1988
Harold Whitaker and John Halas, *Timing for Animation*, Focal Press Ltd, 1981
John Whitney, *Digital Harmony*, Byte Books, 1980

Facilitating Learning with Computer Graphics and Multimedia

Chair
G. Scott Owen, Georgia State University

Panelists
Robert V. Blystone, Trinity University
Valerie A. Miller, Georgia State University
Barbara Mones-Hattal, George Mason University
Jacki Morie, University of Central Florida

Abstract
With the recent advent of inexpensive yet powerful computers, the use of high quality graphics and multimedia systems to facilitate learning is rapidly increasing. This panel will review leading-edge work by focusing on several areas, including computer science, mathematics, biology, and art and design. Each panelist will describe how they currently use computer graphics and/or multimedia and give their view of future applications. Emphasis will be placed on how using these techniques fosters interdisciplinary collaboration both for creating learning environments and for working in these career areas.

Panel Overview
It is becoming increasingly evident that a highly educated and/or skilled work force is necessary for a successful economy, thus the need for more effective means of education/training is extremely important. Recent developments in technology have created the possibility of a paradigm shift in the delivery of information. Such a paradigm shift requires a change in the way we define and understand information exchange, resulting in more effective means of communication. One way in which this change may be effected provides for the possibility of interaction and integration of traditional disciplines to maximize the potential of these emerging technologies.

The objective of this panel is to investigate how this paradigm shift is being implemented in several areas of learning. While the focus is on academic areas of learning, the lessons learned and principles developed will also apply to industrial training and continuing education.

Robert V. Blystone: Computer Graphics in Undergraduate Biology
Biological microscopy is a visual discipline; however, when traditionally used in support of undergraduate learning, microscopy is descriptive, of limited sample size, and two dimensional. By coupling computer graphics with biological microscopy, these limitations can be overcome. At Trinity University, microscope intensive courses such as histology and embryology have been dramatically enhanced through this union of computer graphics and microscopy.

The application of graphics can expand two dimensional microscope images into three and four dimensional data sets. Two approaches have been commonly employed in our lab to develop digital scientific visualizations. The first approach utilizes images of intact structures such as embryos or blood cells. These images are collected under conditions of different age or treatment and then morphed to create visualizations of change through time. As an example, the white blood cell known as a neutrophil demonstrates significant changes in its nuclear shape during its week long existence in the circulation. By collecting images of different aged neutrophils and then morphing these images, a "virtual" time lapse movie can be created to represent the aging process. The second approach requires the image capture of sequential (serial) images (sections) of histological elements. These serial sections can be digitally "glued" back together and projected into three dimensional space. Projections taken at different times can be morphed to show change in three dimensions. As an example, projected serial sections through the developing pituitary of a chick embryo of one age can be morphed with a different aged embryo and, as a consequence, the pituitary can be made to grow in space through time. Animations of these types allow for discussions and inquiry-based activities never before possible in the lab and well as the lecture.

Biology students are generally unskilled in the use of computer graphics; thus, lab exercises had to be carefully organized to lead the students into the transparent use of computer graphics so as not to take time away from the biology subject material. As students request to do undergraduate research utilizing this technology, they must agree to tutor students new to the technology. Further, as students developed quality animations and image analysis procedures, the results could be saved and incorporated into the next semester's class.

Valerie A. Miller: Computer Graphics in Undergraduate Mathematics
The use of graphical images has been traditional in teaching lower level mathematics. However, when a student enters a mathematics class beyond the calculus level the use of images disappears, except for the occasional graph theorist's graph or the plotting of a function. This is rapidly changing, as the use of computer graphics and multimedia in teaching and learning mathematics is a mode of instruction that is becoming an accepted technique in the classroom. With the advent of inexpensive graphing calculators and various software packages that assist in the visualization of 2- and 3-dimensional mathematical concepts, mathematics instruction is beginning to examine avenues of learning other than endless rote calculations.

As an example of mathematical visualization in numerical analysis, we present ways in which this may be used to aid the instruction of iterative techniques for solving the problem $f(x) = 0$. Fractal images are generated based upon the number of iterations needed for convergence for various functions via various methods. These images are then presented simultaneously so that the important aspects of each of these methods, such as the order of convergence and cost per iteration, can be more easily illustrated, compared and, hence, more thoroughly understood. The concept of one method being "more expensive" than another is one that is not easily understood by students and the ability to inspect these "costs" visually is very appealing.

Fractal images that are generated based upon the basins of attraction of a function are also presented as means of illustrating how an iterative method can converge to an unwanted solution. As there are many possible criteria for terminating an iterative process, two different criteria are used to illustrate the subtle effects of a stopping criterion to the student.

Barbara Mones-Hattal: Using Computer Graphics to Teach Art Concepts

Currently, in the art and design area, computer graphics and computer imaging are taught as separate course of study in many schools. At these schools, software tools are used primarily to investigate potential computer applications in 2D and 3D design, animation, and interactive design. The available software tools are designed for the artist to create end-products such as illustrations, design for print, and all forms of animation. To a degree, tool and medium aspects of the technology are very difficult to isolate. However, using these same tools, one can de-emphasize the end-product and more fully explore the design process. It is this process of exploration that underscores the potential of computer graphics and multimedia for facilitated learning.

In this panel session, an overview of projects in the fine and applied arts, from intermediate to advanced level, will be presented. These projects focus on the enhancement of the learning process by utilizing computer graphics or multimedia tools (including paint and draw software, modeling and animation software, stereo display, and virtual reality tools). In some cases, mixing computer skills with other real world objects is included. The acquisition of computer graphics skills is less important in these cases as the purpose is to learn to communicate sensory information effectively and expressively. Important questions involving the potential to use the technologies to pose questions to the participant/viewer about the relationship of physical realities to simulate realities will be discussed.

Computer Graphics and multimedia options have provided many profoundly more efficient ways to learn about basic concepts such as color theories and applications, spatial relationships, and mixed media design. Some new additions to the repertoire of tools available are emerging as the inter-relationship of dimensional design forces us to separate our tools into paint (2D), modeling (3D), or moving images (4D), and encourages us to think of developing our works in new and innovative ways.

Learning both fundamental and sophisticated skills is important as both are required of the artist/designer. But some combination of these skills will enhance the capabilities of the non-artists, or any participant in a collaborative or interdisciplinary research team who wishes to work with artists. As more and more computer graphics and multimedia professionals seek to integrate their skills in order to develop and deliver better products, more effective ways to create and enhance learning environments are needed.

Jacquelyn Ford Morie: Using Computer Graphics to Teach Computer Graphics Concepts to Art Students

It is well known that students learn most effectively with hands on experience that reinforces concepts presented by an instructor through assigned reading or lectures. In computer graphics students can read extensively about a particular technique or algorithm, and still not fully understand it. Providing students with hands on exercises can cause an immediate and intuitive understanding of a complex graphic concept.

Computer Science students studying computer graphics have long been expected to have a "hands on" understanding of graphics algorithms in the context of writing the code to implement them. While this does provide a very thorough and complete understanding of the mechanics of how a given procedure works internally, the overhead of writing code is not the only method to cultivate understanding of graphics concepts. In addition, it may not be the most effective way for a student to understand the rich possibilities inherent in a given graphics technique. For example, writing an L-systems piece of code, while challenging, does little to uncover the wealth and variety of forms it is possible to make with such a system. An interactive program that allows a student to play with making many fern-like objects through L-system rules can, in an hour's time, provide a different and valuable intuitive grasp of the concepts behind the process.

This type of learning is also accessible to a greater range of students. This is especially true for visually oriented students, such as the computer animation students (in both art and film programs) that I have been teaching for the last five years. It is sometimes difficult to get them to read research articles or popular magazine articles about various computer graphics techniques. Give these same students an interactive demo on a computer and it is difficult to get them to go home.

Many such programs are available commercially or through public domain sources. Silicon Graphics provides a series of "Button Fly" demos which illustrate some concepts such as B-splines, environment mapping, ray tracing, and radiosity. There are many interactive fractal generating programs on the market. In conjunction with videos, such as those available from computer graphics suppliers, such as the SIGGRAPH Video Review, these can be valuable tools for understanding. In addition, advanced computer science students can provide interactive instructional programs and tools in the context of a summer class or independent study. These can then be tested and used by students of all disciplines studying computer graphics concepts.

Some very interesting results of using computer graphics to teach Computer Graphics concepts include the following:

- Visually-oriented students get "hooked." Since many of these same students are very "process-oriented" (e.g. they have a keen interest in following through a project from mechanics to the final creation), they become very curious about the process behind computer graphics. Some even enroll in UNIX and computer programming classes. If it is seen to be a means to their specific ends, they will spend the effort to become knowledgeable about all aspect of the process.
- Computer Science students see a greater picture and spend the extra time and effort to make their demo or program "better" than the interactive demo they have been exposed to, increasing the amount of learning they derive from the exercise.

G. Scott Owen: Computer Graphics and Multimedia in Computer Science

While computer graphics has been somewhat used to teach computer graphics, it has not been much used in the rest of the computer science curriculum. This is beginning to change as more instructors are beginning to develop and use graphics software to illustrate different concepts. Graphics programs are used in courses to illustrate such concepts as graphs or tree structures. Algorithm animation has been used at a few select places on workstations but is now becoming available on PC-class machines. Another use of graphics is in programming assignments for introductory classes. Multimedia has just begun to be used and I will discuss one system, HyperGraph, which is used to teach computer graphics. HyperGraph runs on PCs and consists of text, images, animations, and video. HyperGraph is being developed by both artists and computer scientists and the ultimate goal is for it to be used by both types of students for learning aspects of computer graphics.

Panel Summary

The examples given in this panel session of using computer graphics and multimedia for learning should help others who are thinking of incorporating these techniques into their own teaching. Lessons learned can also be incorporated into industrial training systems.

Visualizing Environmental Data Sets

Chair
Theresa Marie Rhyne, Martin Marietta/U.S. EPA Scientific Visualization Center

Panelists:
Kevin J. Hussey, Jet Propulsion Laboratory
Jim McLeod, San Diego Supercomputing Center
Brian Orland, University of Illinois/Landscape Arch.
Mike Stephens, Computer Sciences Corp./U.S. Army Corp of Engineers
Lloyd A. Treinish, IBM T. J. Watson Research Center

This panel session focuses on issues pertaining to visualizing environmental sciences data sets. Here the term "multi-dimensional" refers to the simultaneous display of data sets associated with air quality, water quality and subsurface contaminated soil regions. Issues associated with facilitating collaborative environmental visualization efforts among various research centers are presented. These include high speed networking, data base management, visualization toolkits, and heterogeneous computing platform concerns.

Visualization researchers dealing with the display of environmental sciences data sets present their different viewpoints on controversial issues. Panelists highlight and contrast projects associated with the U.S. Environmental Protection Agency, the U.S. Forestry Service, the Jet Propulsion Laboratory, Scripps Institute of Oceanography, National Aeronautics and Space Administration, U.S. Army Corps of Engineers, IBM T.J. Watson Research Center, University of Illinois, and other government, university, and industry centers working with environmental data sets.

The controversial issues include:

- Visualization of Multi Model data: issues associated with validity of data and approaches to data consolidation for a single visualization are discussed.
- Toolkit Applicability: Each panelists has their own unique viewpoint on how and when toolkits should be used for environmental research projects.
- Renaissance Teams (yes or no??): Some of panelists are at visualization centers which continue to advocate collaborative efforts among researchers, programmers, and artists for environmental visualization projects. Other panelists are involved with efforts focused on developing tools for the direct use by environmental researchers with minimal involvement of Renaissance Teams.
- Research versus Policy Making: Visualization for research efforts versus visualization for regulatory and policy making efforts can yield different end products and toolkit requirements.
- Data Format Standards: Multi dimensional environmental visualization involves the merger of data from multiple sources. This points toward the controversial issue of whether a standard data format for environmental research models which supports heterogeneous computing should be required.
- Data Management: The management of terabyte and gigabyte data sets is one of the critical challenges to environmental research and visualization.

Multi-dimensional data visualizations not only require management of data from satellites but also the maintenance of historical geological data sets.

Theresa Marie Rhyne

The U.S. EPA Scientific Visualization Center serves the U.S. Environmental Protection Agency's community of researchers throughout the United States and collaborates on interagency and university research projects. Within EPA, there are a wide range of interests. The Visualization Center has depicted pollutant transport and deposition in regional domains of the United States, total global ozone distribution, fluid flow around buildings, sedimentation in large bodies of water, subsurface contaminated soil regions, and the molecular chemistry of carcinogens.

Visualization toolkits are used by researchers to examine their data, and intensive 3-day workshop sessions are designed to handle environmental researchers' desires for training in the use of these toolkits. High speed networking concerns and the development of visualization tools to support heterogeneous computing platforms across the Agency are explored using the U.S. EPA's National Environmental Supercomputing Center, located in Bay City, Michigan. Collaborations on multi-dimensional environmental visualizations that display the co-registration of air quality, water quality, and subsurface soil data sets are underway. Issues associated with visualization technology transfer to State and local government environmental protection agencies are being examined.

Jim McLeod

Faced with ever increasing environmental data collection rates (soon approaching terabytes of information a day), the ability to disseminate and analyze these data sets at comparable rates is crucial to the success of future global studies. Visualization plays a vital role. However, visualization should not overshadow the ultimate goal of scientific discovery and global understanding. To this end, visualization should hold equal weight with traditional scientific analysis techniques until its value as a research tool can be measured and evaluated. At the Advanced Scientific Visualization Laboratory in the San Diego Supercomputer Center, we provide a multi-level visualization support service. By utilizing visualization toolkits, at one level, to obtain analytical imagery quickly, researchers and members of our visualization staff (collectively referred to as "Envelope Teams") can determine the feasibility and merit of these visual studies before proceeding to a higher level of custom visualization.

Although this collaborative approach has been tried in the past, it has rarely succeeded scientifically. These visualization efforts have failed to recognize past experiences and techniques to which scientists are accustomed. Therefore, to achieve a smooth migration to new visual analysis methods through the use of computer graphics, visualization tools must involve traditional representation techniques (plots, graphs, and numerical analysis) as well as object rendering. Only after getting these types of tools in the hands of the researchers and maintaining a dialogue to improve the usefulness of these applications will tough visualization issues involving database management and co-registration of data be resolved properly. This communication process will rely heavily on advances being made in high speed networking, which will play an important role in the promising future of environmental visualization.

Kevin J. Hussey

The Jet Propulsion Laboratory's (JPL's) Digital Image Animation Laboratory (DIAL) and the associated Visualization and Earth Science and Applications (VESA) Group have produced a number of notable environmental visualizations over the past 12 years. Visualization topics have encompassed submarine geology in the Monterey Bay to the Ozone "hole" over Antarctica. Scientific data resolutions ranged from one Angstrom to 345 miles. During the course of producing these visualizations several lessons were learned, difficult issues raised and new technology developed. Examples are highlighted below.

A Lesson: No matter how good, flexible, extensible, comprehensive, powerful or expensive your visualization system (hardware and software) is, it will not perform what the scientist wants. You will have to modify some code to make it happen.

An Issue: Remember the book "How To Lie With Statistics" by D. Huff? If you can lie with statistics then just imagine what you could do with Data Visualization! Sometimes very realistic looking visualizations may be "seeing" things that are not supported by the data. Should we do something about this?

Some Technology: Incorporating technology developed and lessons learned, members of the VESA and Image Analysis Systems (IAS) group at JPL are developing Surveyor, a three-dimensional data visualization system which runs in a heterogeneous distributed computing environment. Surveyor is used by scientists and simulators to analyze and animate a three-dimensional "world". The world consists of a variety of three-dimensional data, such as satellite imagery combined with topography information. In addition to rendering of data, Surveyor can retrieve the original data values for selected areas in a three-dimensional rendered scene. This capability is used as a user interface for the analysis of scientific data, such as geological data bases from desktop computers or across a network to allow use of more powerful machines. Surveyor is utilized as the graphical user interface (GUI) for the Caltech/JPL portion of the Casa Gigabit Network Testbed.

Lloyd A. Treinish

Areas of great interest in the environmental sciences today focus on large-scale data processing and analysis of remotely sensed and in situ data from many instruments and the supercomputer-based simulation of dynamic phenomena. Often these studies involve the integration of the observations with simulations for the creation of empirical models, using the acquired data as input. The structure of these data may be point or sparse, uniformly or irregularly meshed, in rectilinear or curvilinear coordinates, and will consist of many parameters. Visualization is key in understanding these data sets.

Effective visual examination also requires advances in data management. There is no consensus today about solutions to problems involving: addressing inconsistencies and irregularities associated with observational data, maintenance of a connection between image representations of the data and the data themselves, integration of observational data with output from computer models, and scalability to potentially very large size (i.e., greater than one terabyte). One aspect of the data management problem is the ability to uniformly support data of disparate structure, preserve the fidelity of the data (e.g., by noting missing data and original grid resolution), and provide the ability to define coordinate systems onto which different data may be registered in space and time. For visualization tools, data management is important in matching a class of data models to the structure of scientific data and determining how such data is used (e.g., qualitative three-dimensional displays AND precise, analysis and quantitative presentations). In work with disparate data, there is a need to support correlative data analysis providing a common basis for the examination of different data sets in the same way at the same time and providing multiple ways to study the same or different data.

Brian Orland

The author directs a research program in environmental visualization and perception. Research ranges from modeling fishing and ski resort choices, to scenic and policy impacts of forest insect damage. Evaluations are based on calibrated visualizations of forest conditions. Few participants are expert in all scientific areas represented in a forest ecological system. Thus, visualizations must not assume ability to deal with visual abstractions of particular scientific data. Focus is on more realistic visual images than typically found in scientific visualization. A second concern is that a forest scientist, manager, or interested citizen should do more than just respond to pre-planned situations. Our user makes modifications to the visualization and obtains immediate feedback on the implications for data or models.

Software development supports this research and addresses a number of inadequacies in current natural resource scientific visualizations:

(1) Realistic, "concrete," representations of biological models and data support interpretation and evaluation by non- specialists while retaining their validity as data representations.

(2) Data entries and model parameters are manipulated via the visualization and the database is continually updated. Users can participate in forest operations, modify scientific assumptions by direct mouse-click operations, and continually monitor their progress through data summary tables.

(3) Visualizations are truly interactive. Advances in time, changes in viewpoint, and model parameters are viewed immediately — the design criterion is a maximum 3 second delay for display update.

Current work will extend visualizations from thousands of trees to millions of trees, and will distribute model computing to faster machines — such as NCSA's CM5 Connection Machine.

Mike Stephens

The scientific visualization center (SVC) at the U.S. Army Corp of Engineers'' Waterways Experiment Station (WES) assists research engineers and scientists in the six major labs which comprise WES. Since environmental research involves a large number of highly coupled, interdependent processes, these efforts are spread throughout the Hydraulics Lab, Coastal Engineering Research Center, Information Technology Lab, and Environment Lab. The SVC advocates a technology transfer approach. Project researchers and their staffs are exposed to visualization alternatives and then determine appropriate methodologies for displaying their own data sets. The SVC has been active in environmental studies from simple conceptual animations of how a newly designed dredge head will reduce the number of turtles adversely effected during dredging operations to complex interactions of ground water flows on sub- surface contaminants from leaking storage tanks.

Several environmental projects involve not only teams from different WES labs, but also include other research agencies. A project on the management of the United States' largest estuary, the Chesapeake Bay, has research teams from WES hydraulics and environment labs as well as the U.S. EPA and several universities. Perhaps it is the highly interdependent nature of environmental processes that make them particularly challenging to visualization teams to offer solutions which aid researchers to further their understanding. The Chesapeake Bay effort has used visualization in almost every aspect of the project. Techniques for evaluating and editing finite element meshes used in computing hydrodynamics models of the bay were developed. The results from these hydrodynamics models were visualized. Once the models were validated from physical measurements they were used in turn by the water quality group which is concerned with the viability of the bay and the effects that different management strategies have on the bay and the long term results of these newly proposed strategies. This involved the examination of 22 interrelated parameters.

How to Lie and Confuse with Visualization

Chair
Nahum D. Gershon, The MITRE Corporation

Panelists:
James M. Coggins, University of North Carolina at Chapel Hill
Paul R. Edholm, The University Hospital of Linkoping, Sweden
Al Globus,Computer Sciences Corporation at NASA Ames Research Center
Vilayanur S. Ramachandran, University of California at San Diego,

Abstract:
As in other fields such as statistics and cartography, it is also possible to misrepresent data in visualization. Most of the time, people do it unintentionally and it goes unnoticed. But traps await the unwary. The Panel "How to Lie and Confuse with Visualization" will discuss this issue and educate the visualization and computer graphics community about these potential traps. Topics to be discussed in this panel include the use of color, interpolation, smoothing, boundaries, and shading. The panel and the audience will also debate whether there are ways to judge the degree of "lying" in visualization, and how to prevent inadvertent misrepresentations from happening. The panel and the audience will come up with recommendations for "dos" and "don'ts" for the process of creating faithful visualizations. During the discussions and the debates, the panelists and the audience will present many examples of data misrepresentations taken from science, medicine, and art.

Other points include the question if lying with visualization is evil or good and necessary and appropriate in many situations. How much is lying with visualization application-dependent ? Does visual perception have problems or is it very efficient? This point may affect the mechanisms underlying the perception of misrepresentation of visualized data.

The audience and the public have been encouraged to submit samples of slides and video material illustrating visualization "lies." In addition, contributors could bring their samples directly to the session.

Nahum Gershon—"How to Lie and Confuse with Visualization"
Inappropriate use of color could reduce the effectiveness of the results of the visualization process. Color scales without sufficient contrasting regions in the range of values where the data is varying could mask these data variations from the final display. Data structures with shapes familiar to our perceptual system could also create false impressions of the data. For example, if a structure in the data appears as to occlude another it could be perceived as having a higher value than its surrounding area. Colors such as light green and blue are usually perceived as background colors in everyday life. In data visual representation, these background colors could be perceived as representing lower values especially if these regions are large. Crowded color scales, on the other hand, could give too much detail, preventing effective perception of the general trends.

Interpolation (the process used to enlarge the display of small spatial data sets and to eliminate the appearance of the visual annoyance of large pixels) could create the false impression of high spatial resolution. Smoothing (the operation used to eliminate random noise in the data) could make small details disappear.

James M. Coggins —"Lying Toward the Truth"
The challenge of the visualization specialist is to cleverly communicate the truth. Truth can be very, very complicated. Revealing the truth a layer at a time is a reasonable strategy for managing the complexity of the whole truth. Half-truths or even outright fabrications are appropriate and necessary mechanisms for sneaking up on the truth.

Visualization provides new ways to present half-truths that advance understanding toward the whole truth. However, when we begin to rely on the half-truths as if they were the whole truth, crucial aspects of the intended communication can be lost. Hard-edged surfaces of discrete objects make beautiful visualizations but hide the often arbitrary criteria used to form the surfaces. The essential uncertainty in the existence or location of boundaries is masked. Naive interpretations of such visualizations may yield faulty decisions.

Mechanisms for visualizing uncertainty are required to communicate error bars on the measurements that are visualized. Whether the hard-edged visualization is adequate is a question for the application domain client, not for the visualization specialist nor for the visualization system. The visualization system must provide the client with error bars: a sense of when to believe the beautiful, hard-edged rendering and when to disregard or disbelieve it. Examples of how error bars can be visualized will be illustrated. Discussion of other alternatives will be solicited.

Paul R. Edholm —"How the Visual System Interprets Images"
The task of the visual system is to give to the conscious mind a mental representation of the relevant features of the external world. From the flat two-dimensional (2-D) images on the retina, the visual system reconstructs the external three-dimensional (3-D) scene. This is in theory impossible because there are so many 3-D scenes which could yield the same 2-D image. So, the visual system has to try to select the right solution from all the possible solutions. This solution is then presented to our conscious mind as the external world. The mechanisms in the intuitive and unconscious interpretation system of the visual system are in their most part unknown. We only know that they are very complex and highly sophisticated. They probably constitute the most complex structure in the world. But when we see, we are not aware of this. We experience (the illusion) that what we see is a direct perception of the external world. In reality, what we see is a very sophisticated guesswork of the interpretation system.
Boundaries: The boundaries are the most important structures in the interpretation of a picture. Thus, the visual system is more or less developed as a boundary detector and it is a poor judge of density values in a picture. I will discuss lateral inhibition, Mach bands, and illusions caused by lateral inhibition. I will also describe the perception

of boundaries and various kinds of boundaries. Boundaries in "natural" images (produced on the retina by real scenes) and images of other kinds (produced on the retina by different artificial means) will be discussed. This will include illusory contours, statistical boundaries, boundaries in X-rays, edge boundaries, boundaries produced by curved interfaces, lamellar boundaries, and illusions caused by curved interfaces combined with lamellas.

Reconstruction of 3-D Scenes from 2-D Images: The clues used by the visual system to reconstruct three-dimensional scenes from two-dimensional images, impossible figures, and 3-D illusions will be discussed.

Al Globus —"Thirteen Ways to Say Nothing with Scientific Visualization"

Scientific visualization should be used to produce beautiful images. Those not properly initiated into the mysteries of visualization research often fail to appreciate the artistic qualities our pictures. For example, scientists will frequently use visualization to needlessly understand their data. I will describe a number of effective techniques to confound such pernicious activity. The audience and the panel will be solicited for additional techniques. The 13 ways are:

1. Never Include a Color Legend
2. Avoid Annotation
3. Never Mention Error Characteristics
4. When in Doubt, Smoothe
5. Avoid Provide Performance Data
6. Quietly Use Stop-Frame Video Techniques

Faithful adherence to the rest of the rules will help avoid tedious debugging of software that already produces pretty pictures.

7. Never Learn Anything About the Data or Scientific Discipline
8. Never Compare Your Results with Other Visualization Techniques
9. Avoid Visualization Systems
10. Never Cite References for the Data
11. Claim Generality but Show Results from a Single Data Set
12. Use Viewing Angle to Hide Blemishes
13. 'This is easily extended to 3-D'

Vilayanur S. Ramachandran—"What Could be Learned from Perception?"

Computers have provided us with new ways of creating visual images from abstract and non- abstract data. Reaching out to the fields of visual physiology, psychophysics, and cognitive psychology could not only explain why human vision is so efficient, but also how to create better images and what could be the limitations of particular representations. The relevance of the knowledge acquired from perception research to the process of creating faithful visualizations will be described. Examples are the areas of stereopsis, perception of transparency, derivation of shape from shading and illusory contours, and color. In particular, I will discuss and illustrate the possible trap of locating the light source in the "wrong" location in simulating shading. I will then specify the situations where the perception of shape from shading could be affected by the location of the light source.

Afterword

Data could be misrepresented in visualization quite effortlessly and inadvertently. The main culprits are perception's complex and intricate nature and the varied experiences of each human being that could make a picture mean different things to different people. Becoming aware of how to lie and confuse with visualization could teach us how to prevent it from happening or at least how to reduce its occurrences.

References

D. Bailey, "Twelve Ways to Fool the Masses When Giving Performance Results on Parallel Computers," *Supercomputer Review*, Aug. 1991, pp. 54-55.

J. M. Coggins, "Computer-Aided Object Definitions from Digital Images," *Proceedings* of Joint U.S.-Scandinavian Symposium on Future Directions of Computer-Aided Radiotherapy, Radiation Research Program, Division of Cancer Treatment, National Cancer Institute, pp. 79-97.

P. Edholm, "Boundaries in the Radiographic Image. I. General Principles for Perception of Boundaries and Their Application to the Image." *Acta Radiol.* 22, 457 (1981).

P. Edholm, "Boundaries in the Radiographic Image. II. Principles for the Representation of the Object as Boundary Circuits in the Image." *Acta Radiol.* 23, 161 (1982).

N. D. Gershon, "Enhanced Visualization of Multi-Dimensional Structures. Applications in Positron Emission Tomography and Climate Data", *Proceedings* of Visualization '91, pp. 188- 192, IEEE Computer Society Press, Washington, DC (1991).

N. D. Gershon, "How to Lie and Confuse with Visualization," *IEEE Computer Graphics and Applications*, pp. 102-103, January 1993.

A. Globus, "13 Ways to Say Nothing with Scientific Visualization," Report RNR- 92-006 NAS, Applied Research Branch MS T045-1, NASA Ames Research Center Moffett Field, CA 94035-1000.

S. M. Pizer, J. M. Coggins and C. A. Burbeck, "Formation of Image Objects in Human Vision," *Computer-Aided Radiology* (invited paper in *Proceedings* of 1991 Conference on Computer- Aided Radiology), Springer-Verlag, Berlin, 1991, pp. 535-542.

V. S. Ramachandran, "Perception of Shape from Shading," *Scientific American*, 269, 76-83 (1988).

V. S. Ramachandran, "Visual Perception in Humans and Machines," in *AI and the Eye*, A. Blake, and T. Troscianko, (eds), pp. 21-77, John Wiley & Sons, New York (1990).

The Applications of Evolutionary and Biological Processes to Computer Art and Animation

Panel Organizer: William Latham, IBM UK Scientific Centre
Chair: George Joblove, Industrial Light & Magic

Panel Members:
William Latham. IBM UK Scientific Centre
Karl Sims. Thinking Machines Corp.
Stephen Todd. IBM UK Scientific Centre
Michael Tolson. Xaos Inc.

Abstract

The panel will discuss new techniques for evolving art designs which are based on "Evolution and Biological processes" from the natural world. In particular techniques such a mutation, breeding and selection, marriage, and rules for artificial life animations are discussed. In addition to addressing the advantages and disadvantages of using these techniques, the panel will also discuss their effectiveness as construction and user interface tools for the artist making images, designs and animations.

Background

In general "artistic" computer graphics involves detailed analytical knowledge to specify a design or animation. Often a designer when setting up a geometric structure realizes the huge possible number of variations that can be produced by changing the parameters. Invariably these variations are not explored, even though a variation may be better than the analytically specified design. The reason for this is the vastness of parameter space and time taken to explore it.

Computer Evolutionary techniques allow a more systematic and intuitive exploration of parameter and structure space which allows the user to evolve artistic designs and animations through a method of random mutation, breeding and selection. This allows not just rapid sampling of possible variations but allows a directional and purposeful exploration of variations. This exploration often uncovers possibilities which are beyond the artists'/designers' powers of human visualization, and are arguably beyond their imagination.

These techniques are based on the evolutionary and biological processes in the natural world. They have a major advantages in terms of user interface as they allow the user to interact in a "non-analytic" way intuitively with the computer, so that highly complex models can be evolved. The use of these techniques creates a new style of user interface where the role of the artist can be separated in two: First as "creator" of the Evolutionary system and then as "selector."

This exploits techniques such as simulated annealing, steepest ascent and Monte Carlo optimization. Genetic algorithms borrow from the biological models of genotype (encoding the form), phenotype (expression of coded form), Mutation and sexual reproduction, and selection for simulating evolution. The most exciting effects have been created when these evolutionary techniques have been merged with their own "home grown" growth systems such as Latham and Todd's "Form Grow" or Sim's "Growing Equations." The resulting films are very organic and portray virtual computer worlds operating under alternative evolutionary systems. As the cinema and Virtual Reality's insatiable appetite for the extraordinary increases, these techniques currently being used by a few key experimenters will find their way into the popular domain.

Panel Goal and Issues

The purpose of the panel is to propose the new use of evolutionary and biological concepts as methods for making art and animations and show the unique effects that can be achieved, and try and define their advantages and differences compared to other techniques for artistic design.

The panel will identify the differences in techniques between the work Sims, Latham and Todd and Tolson. Specific techniques such as "structure mutation," "equation mutation," and "automatic offspring selection" will be discussed. Mike Tolson will contrast these techniques with his work with growing brushed patterns using evolution genetic fitness algorithms which use no human user selection.

Having laid down the technical overview, the following types of questions will try to be answered:

* In creating alternative evolutions is this just another manifestation of man's innate desire to create new life as a "pseudo god," or is it a parody of man's manipulation of the natural world through modern technology?
* What is the role of the artist: to be both the "creator" of the system and be the "selector." If these roles are done to by two people who then is the author of the artwork?
* Mutation techniques use randomness, does this make them difficult to use with exact control? or is it possible to mix mutation with analytical techniques?
* In building menu options such as "kill," "breed," or "marry" and applying them to artificial life that often to appear to have a life of their own, does this raise any morality questions?
* What are the future possibilities of these techniques in scientific visualization?
* Latham proposes an Evolution Virtual Reality where the artist would become a "gardener" in an accelerated evolutionary world growing and breeding "giant virtual pumpkins" in "living sculpture gardens," a kind of modern day "Garden of Earthly Delights" (Hieronymous Bosch). What other applications of these techniques in VR could exist in the future.
* Genetic Algorithms have been around for 15 years, is Evolutionary Art just a novel application? What is new?
* When the artist subjectively selects mutations for breeding what criteria are they using? If this can be defined would it not be possible to write a piece of software that could replace the artist?
* Can one label the products of computer evolution art? And if one does, surely then the products of nature should be labeled "art" also in that they also went through an evolutionary process. As Computer Art and Animation become more natural do they become less artistic as the human element becomes less apparent.

William Latham

William Latham will focus on the artistic side of the work and discuss the user interface of Mutator program and outline it's successes and weaknesses as a method for making art.

Latham describes using the Mutator program to be a little like being a gardener breeding, selecting, pruning and marrying forms. The evolved forms look like strange organic sculptural fruits. In the

Mutator program, unlike the natural world, the "natural selection" process is replaced with "aesthetic selection controlled by the artist," so that the process of making art is an explorative evolution steered by the results of aesthetic choices. The evolutionary process helps the artist navigate in an infinite multi-dimensional parameter and structure space to find artistic forms.

Latham argues that when using the "mutator" program the computer ceases to be like a tool but is more like a creative partner where the artist is continually surprised by the results which are automatically produced. This close interaction between artist and machine produces unexpected and strange results.

Recent work has involved inventing "Life Cycle" rules to create organic animations showing chains of living, breeding and dead mutations gradually forming vast colonial formations.

Latham will discuss the problems of setting up "artificial animations" and in particular the "Life Cycle" rules. He will also identify some of the artistic issues in the work such as randomness, the balance of power between artist and machine, displaying the artworks in the gallery world and its position in 20th Century Modern art.

Stephen Todd

Stephen Todd will discuss the program Mutator as a user interface tool. Mutator is a tool to assist a user in a search of a multidimensional parameter space. The computer makes moves in space and displays the results of these moves, and the user selects the results that are liked.

Mutator can be looked at as a computer implementation of the process of natural selection, with the mutation process performed by the computer and the selection by the user. Alternatively, it can be looked at as an optimization process, with the cost function provided by the user.

Mutator gives the user a variety of ways to control the movement in parameter space.

- The simplest is random mutation of the parameters.
- The simple random search can be directed by judging some of the available choices as 'good' or 'bad.' These judgments set up a direction of movement, similar to the use of hill climbing in optimization.
- Finally, preferred objects can be 'married' to create new objects that use a mixture of the parameters of the parents.

The augmentation of random mutation by judgment and marriage considerably enhances the speed and effectiveness of Mutator in reaching interesting areas of the search space.

The primary advantage of Mutator is that it permits the user to search using *subjective* decisions. This contrasts with more conventional user interface tools, which make it easier for the user to carry out *analytic* decisions.

Todd will discuss the potential for Mutator as an interface in other applications such as scientific visualization, economic modeling and the generation of 'identikit' pictures by witnesses.

Karl Sims

Karl Sims will present several applications of interactive evolution. The evolutionary mechanisms of variation and selection are used to "evolve" equations used by various procedural models for computer graphics and animation. The following examples will be briefly discussed and results from each will be shown:

- Procedurally generated pictures and textures
- 3D objects defined by parametric equations
- Dynamical systems described by differential equations.

Each uses hierarchical lisp expressions as "genotypes" to define arbitrary equations which are evaluated to create resulting "phenotypes." The equations and their corresponding phenotypes can be mutated, mated, and interactively selected to search "hyperspaces" of possible results. A comparison between evolving values of parameter sets and evolving arbitrary length equations will be made.

"Genetic cross dissolves" can be used to create smooth interpolations between evolved entities. Their use in creating the animation "Primordial Dance" will be described.

It is proposed that these methods have potential as powerful tools for exploring procedural models and achieving flexible complexity with a minimum of user input and knowledge of details. Complex equations can be efficiently found that might not be easily designed or even understood by humans alone.

Mike Tolson

Michael Tolson will talk about his recent work evolutionary and biological techniques: He is creating "eco systems" of many simple "animals" which are visualized by brush strokes. Their behaviors are controlled by neural networks (their "brains") which are a part of the genotype and can be evolved. The animals use up energy by moving and can gain energy by performing appropriately, for example moving towards a light. They can interact with their environment in various ways and can achieve "biological" effects such as phototropism and reaction-diffusion like systems. He is not using interactive selection, but instead the more traditional genetic algorithm approach with defined fitness functions. He is however, more interested in creating "art" than in the scientific details of the process.

Summary Statement

The aim of the panel is to give an overview by leading exponents, contextulise their work, to provoke discussion and be provoked.

References

Evolutionary Art and Computers. Stephen Todd and William Latham. Academic Press. ISBN 0-12-437185-X.

"Interactive Evolution of Dynamical Systems," *Proceedings* of the First European Conference on Artificial Life, Paris, Dec. 11-13, 1991, K.Sims.

The Blindwatchmaker. Richard Dawkins. Longmans Scientific and Technical.

Computer Sculpture Design and Animation." S.Todd, W.Latham,P.Hughes. *The Journal of Visualisation and Computer Animation*. Vol 2. 1991. 98-1. Wiley.

"Mutator, A Subjective Interface for Evolution of Computer Sculptures." IBM UKSC Report. No.248. W.Latham and S.Todd.

"Evolution by Aesthetics." M.Haggerty. *IEEE Computer Graphics and Applications*. March 1991..

"Surreal Art or Artificial Life," *Proceedings* of the First European Conference on Artificial Life, Paris, Dec. 11-13, 1991, S.Todd and W.Latham.

"Artificial Evolution for Computer Graphics," *Computer Graphics*, Vol.25, No.4, July 1991, pp.319-328, K.Sims.

Genetic Algorithm - in Search, Optimisation and Machine Learning. D.Goldberg. Addison Wesley. 1989.

Adaption in Natural and Artificial Systems. J.Holland. University of Michegan Press 1975.

Urban Tech-Gap: How the Museum/University Liaisons Propose to Create a Learning Ladder for Visual Literacy

Chair: Richard Navin, Brooklyn College, City University of New York

Panelists:
Lynn Holden, Carnegie Mellon University
Edward Wagner, The Franklin Institute Science Museum
Robert Carlson, Director, Tech 2000: Gallery of Interactive Multimedia
Michael McGetrick, Brooklyn College

Summary

Universities and public education are putting the creative means of professional multimedia production into their own hands. Science and industry museums engaging media as producers now offer serious instructional visualizations through creative hands-on components drawing visitors into active participation. The distance is closing between museums' high-calibre productions and educational usage through open electronic toolboxes in the individual classroom. Yet a severe dislocation exists between the producing centers and the decay in their immediate environment — the inner city.

Collaboration bridges strengths and resources from two methodologically different environments. This panel seeks to place a bridge between the differences of these two professional worlds and develop a ladder for visual literacy as a common domain.

Issues Under Discussionn

- The finest science and industry museums exist in midst of the worst financially-blighted urban centers the most abandoned urban school districts. Are minority-dominated, overcrowded classrooms unable to come to grips with, or understand in a rudimentary fashion, rapidly advancing technology linguistics — particularly when their computer labs are crippled or virtually non-existent? Tax-levied education faces increasing draconian budgets for three to five year. How can tools and multiculturalism be integrated with the means at hand?
- What curriculum bridges can be created in advance of exhibitions? What social components should determine their final form? Between usage and exhibit production values, can a more sensitive awareness enter "opening the architecture" and creating tools that bridge the discontinuity between high-level productions finished in a museum "Hollywood" setting and real learning? What is needed are forms that transmute content through "the language of the street" — forms understood by the average (and deprived) urban public school classroom and those students snarled in its chaos.
- Why are most high-quality, master computer centers distanced (safely) from the largest inner city school districts? Should high-tech be physically downloaded into some form for urban equilibration following the marketing of GI Joe, comics, pop-up books, music or rap oriented materials? This is such an untried avenue that we cannot even begin to believe we approach "separate but equal" social segregation.
- Downloading: High production values in themselves induce passivity (i.e. the "Carl Sagan" approach gives us very high, but passive values on the edu-tainment meter). Few projects of meaningful application are developed in the argot of the children watching. Nothing induces vocational interest or offers access to a science career in a designated spot within the viewer's interactive participation and behavior modeling. Should productions including some staffing role that incorporates "human tools" drawn directly from the disenfranchised and underprivileged where we deem to do good?
- Do exhibition, production-value approaches too easily adopt a TV cultural fast-food pace and too speedily send viewers on their way?

At the museum level, sheer numbers of viewers and the fifteen minute message are determined by large attendance traffic necessities. At the schools, amateurism often mars the connectivity of video, hypertext, and programming, detouring the expectations of visually astute and sophisticated youths. Educators and museum program designers on this panel offer real links between visitor/learners and serious instruction. Written materials can support strong visuals and supply stick to the ribs knowledge in the wake of video flash. The dialogue between panelists contrasts the style and formatting problems of exhibiting and instructing and offers a "how to" on establishing usage between museum educational institutions and visually attractive teaching modules, while still downloading qualified and deep instruction without losing the sweep and verve of highly attractive visualization.

Lynn Holden: The great pressing need is how to link and move forward the existing elements of the multimedia matrix. The human, hardware and software components all exist, but they are dislocated and without adequate resources and financial support from government, industry and local communities/institutions.

If there is not a concerted effort, cooperatively between universities, non-profit institutions and industries to set high quality standards in new applications and products, the near-term ramifications will be disastrous for us as professionals, for our organizations and for out culture!

We must create and support an effective, functional network for developers and creative individuals working on new multimedia and interdisciplinary project, and facilitate completing and making visible actual examples which address short and long term issues.

Ed Wagner: The Cutting Edge Gallery presents new technologies and products to a wide range and intellects. As gallery coordinator I procure these devices and have to quickly learn how to operate them and understand the technology, then present this information to the visitor. The challenge is to take complex scientific information and translate it into a form that is comprehensible to the lay person. The Gallery, like other areas within the Franklin Institute, transfers this knowledge to wide range of visitors while in an informal setting.

Richard Navin: One clearly sees across the audiences of all national conferences a lack of color... minorities are no where to be bound in a significant population as makers of multimedia except as strategically marketed talking head so news shows and sit-coms. City University of New York is woefully behind in adapting the media-designing computer to its constituency — the representation student population. In some small but significant way Image and Communications Projects now place students in design roles with government-aligned agencies. Our next step from the advantaged position of a new Media Center created by a new administration is to begin electronic publishing. We have reasonable found self esteem. We now need to advance into an articulate electronic venacular.

Virtual Reality and Computer Graphics Programming

Chair:
Bob C. Liang, IBM T. J. Watson Research Lab

Panelists:
William Bricken, University of Washington
Peter Cornwell, Division, Inc.
Bryan Lewis, IBM T. J. Watson Research Lab
Ken Pimental, Sense8 Corporation
Michael J. Zyda, Naval Postgraduate School

Virtual Reality provides a multi-sensor 3D human machine interface. VR applications requires system software interface to various input/output devices, e.g. tracker, glove, head mount display, sound, speech recognition, etc., and a programming environment for building and interacting with the virtual world.

We are addressing software issues of Virtual Reality related to computer graphics: the building of the virtual world, and the management of its underlying data structure; the communication software issues in building a cooperative virtual world environment; and the issues in programming the interaction in the virtual world.

Michael J. Zyda: The Software Required for the Computer Generation of Virtual Environments

The first phase of virtual world development has focused on the novel hardware (3D input and 3D output) and the "cool" graphics demo. The second phase of virtual world development will be to focus in on the more significant part of the problem, the software bed underlying "real" applications. The focus of this talk is on the software required to support large scale, networked, multi-party virtual environments. We discuss navigation (virtual camera view point control and its coupling to real-time, hidden surface elimination), interaction (software for constructing a dialogue from the inputs read from our devices and for applying that dialogue to display changes), communication (software for passing changes in the world model to other players on the network, and software for allowing the entry of previously undescribed players into the system), autonomy (software for playing autonomous agents in our virtual world against interactive players), scripting (software for recording, playing back and multi-tracking previous play against live or autonomous players, with autonomy provided for departures from the recorded script), and hypermedia integration (software for integrating hypermedia data-audio, compressed video, with embedded links —into our geometrically described virtual world). All of this software serves as the base for the fully detailed, fully interactive, seamless environment of the third phase of virtual world development.

William Bricken: Virtual Reality is Not a Simulation of Physical Reality

One of the weakest aspects of current software tools for VR is that designers are bringing the assumptive baggage of the world of mass into the digital world, undermining the essential qualities of the virtual. Information is not mass; meaning is *constructed* in the cognitive domain. Psychology is the Physics of VR. In building virtual worlds, we are continually discovering that they are strongly counter-intuitive, that our training as physical beings obstructs our use of the imaginary realm.

The greatest design challenge for VR tools is mediating between physical sensation and cognitive construction. VR software must directly resolve the mind-body duality which plagues both Western philosophy and computer languages. VR calls for a philosophy of *immaterial realism*. VR doesn't matter, it informs.

I will briefly describe a new generation of software tools which emphasize virtual rather than physical modeling concepts. VR tools are situated, pluralistic, synesthetic, paradoxical, and most importantly, autonomous. Their programming techniques include behavioral specification (entity-based models), inconsistency maintenance (imaginary booleans), possibility calculi (set functions), relaxation (satisfying solutions), experiential mathematics (spatial computation), participatory programming (inclusive local parallelism) and emergence (realtime non-linear dynamics).

Peter J. Cornwell

Division Inc. is a VR systems company which has supplied hardware, software and integrated systems products to application developers and end users worldwide for over three years.

Research and commercial applications of VR in the fields of pharmaceutical engineering and marketing, landscape architecture, industrial furnishing and lighting will be described and the evolution of installed hardware and software briefly reviewed.

The important trends arising from these installations will then be identified with particular emphasis on:

1. protection of investment in applications development through portability across a range of different vendor and performance hardware configurations.
2. interfacing VR facilities with existing computer systems and software

Finally, current and proposed developments to address the evolving VR marketplace will be covered.

Bryan Lewis : Software Architecture

Virtual Reality applications are difficult to build, involving multiple simultaneous input and output devices, complex graphics, and dynamic simulations, all of which have to cooperate in real time. If a simulation expert wishes to imbed a complex simulation or model into a virtual environment, one approach is to add functionality one call at a time to the existing simulation code. This appears to be the easy way to get started. But simulations are already quite complex, and adding code for multiple devices and multiple machines can become unwieldy. The problem is made worse by the fact that simulation experts generally do not have the time or inclination to learn a book full of user interface calls.

A better approach is a "minimally invasive" software architecture that allows the simulation to be connected to the rest of the world without surgery. Such an architecture hides the complexities of connecting the simulation to other modules on other machines. This provides the foundation for a useful tool kit; to that must be added a kit of ready-to-reuse modules (for various devices and interaction techniques), and graphical tools to assist in building the world.

Kenneth Pimentel : 3D Graphics programming simplification
I would discuss how the process of interacting and creating a 3D simulation can be drastically simplified by using an object-oriented graphics programming language. I will also mention how this approach provides the utmost in flexibility which is critical in an emerging field. No one knows the "right way" to do things. Experimentation, rapid prototyping, and flexibility are key.

I will contrast the benefits and trade-offs of a graphic oriented interface versus a programming interface for the creation of virtual worlds.

I will bring a video tape showing various types of simulations created using WorldToolKIt. This will show the audience the diversity of applications already created.

Ubiquitous Computing and Augmented Reality

Chair:
Rich Gold, Xerox PARC

Panelists:
Bill Buxton, University of Toronto & Xerox PARC
Steve Feiner, Columbia University
Chris Schmandt, M.I.T. Media Labs
Pierre Wellner, Cambridge University & EuroPARC
Mark Weiser, Xerox PARC

" The door refused to open. It said, 'Five cents, please.' He searched his pockets. No more coins; nothing. "I'll pay you tomorrow,' he told the door. Again he tried the knob. Again it remained locked tight. 'What I pay you,' he informed it, 'is in the nature of a gratuity; I don't have to pay you.' 'I think otherwise,' the door said. 'Look in the purchase contract you signed when you bought this conapt.' "

-Philip K. Dick, from "Ubik"

Panel Background
Ubiquitous Computing is a radical alternative to the desktop and virtual reality models of computing. It turns these models inside out: instead of using computers to simulate or replace our common physical space, computers are embedded invisibly and directly into the real world. Everyday objects and our normal activities become the I/O to this highly sensuous and reactive environment. Objects are aware of and can respond to the location, state and activities of other objects in the world, both animate and inanimate. The implications are important: computing should be part of our everyday existence rather than isolated (and isolating) on a desktop; of equal importance, computer-based systems can take advantage of, and be compatible with, the rich environments in which we live. The poverty of the workstation, with its limited display, array of keys and single, simple pointing device (bad for the eyes, bad for the back, bad for the marriage) becomes clear when designers try to integrate it into our complex social and physical environments.

Panel Goals
The July 1993 issue of the Communications of the A.C.M., co-edited by Pierre Wellner, Wendy Mackay and Rich Gold, explores the research fields collectively known as Augmented Environments. This panel brings together five visionaries within this field to look at its implications and future. Each member is an expert within a different domain including ubiquitous video, projected reality, augmented reality, ubiquitous audio and infrastructure. Combined, these areas create a powerful new way of interacting and living within a computational environment. After a brief overview, each panel member will present a perspective on their own work in the field. Following these presentations panel members will have an opportunity to discuss the over-arching features of ubiquitous computing and answer questions from the audience.

Bill Buxton
Work in UbiComp has been paralleled by work in Video "Mediaspaces." Our position is that in such Mediaspaces, it is just as inappropriate to channel all of one's video interactions through a single camera/monitor pair; as it is to channel all of one's computational activity through a single keyboard, mouse and display. This leads us to the notion of Ubiquitous Video, which parallels that of Ubiquitous Computing.

When UbiComp and UbiVid converge, all of the transducers used for human-human interaction are candidates for human-computer interaction. Hence, for example, your computer can "see" you using the same camera that you use for videoconferencing, and it can hear you using the same microphone that you use for teleconferencing. These are not just changes in the source of input, however. They imply important changes in the nature of information available. Traditional human-computer dialogues have focused on foreground "conversational" interaction. Much of what will emerge from this Ubiquitous Media environment is much more similar to remote sensing. What is provided is background information about the context in which conversational interactions take place. It is this bimodal foreground/background interaction which we will pursue in our presentation.

Steve Feiner
Virtual reality systems use 3D graphics and other media to replace much or all of the user's interaction with the real world. In contrast, augmented reality systems supplement the user's view of the real world. By adding to, rather than replacing, what we experience, an

augmented reality can annotate the real world with additional information, such as descriptions of interesting features or instructions for performing physical tasks. For this supplementary material to be as effective as possible, we believe that it should not be created in advance, but should rather be designed on the fly using knowledge-based techniques that take into account the specific information to be communicated and the state of the world and user.

To test our ideas, we have been building an experimental, knowledge-based, augmented reality system that uses a see-through head-mounted display to overlay a complementary virtual world on the user's view of the real world. In a simple equipment maintenance domain that we have developed, the virtual world includes 3D representations of actual physical objects and virtual metaobjects such as arrows, textual callouts, and leader lines. A knowledge-based graphics component designs the overlaid information as it is presented. The design is based on a description of the information to be communicated and on data from sensors that track the position and orientation of the user and selected objects.

Chris Schmandt
Communication is what needs to be ubiquitous. "Computing" is what will provide the backbone to allow transparent mobility, and, perhaps more important, filter or control access so we do not get overwhelmed with all the stuff that comes down the pipes at us. Communication is even more about voice than it is about text and/or graphics. Ubiquitous computing has to handle voice! Voice interfaces may let me wander around my office and interact with computers through a window larger than that provided by a 19 inch display. Voice is how my environment will communicate with me as I move through it in the course of a work day. Voice is what will let us miniaturize devices below the sizes of buttons and displays.

The pocket phone provides a highly mobile computer terminal with today's technology. Telephone access to email, voicemail, my calendar and my rolodex, as well as various online information services, has already begun to change the way my group works. Wireless digital telephone networks using low powered pocket transceivers are inevitable. Although these are usually touted as "Personal" Communication Networks, telephone service providers have never understood that personalization requires dynamic decisions about how to route communication, and cannot be expressed as static routing tables in a telephone switch. Here's where the "computing" comes in — deciding which calls (or which email for that matter) will make it through my filters to ring the phone in my pocket.

Mark Weiser
A technological response to the challenge of making life better is to radically reconceive the computer around people's natural activity. Two key points are: people live in an environmental surround, using space, muscle memory, 3-D body surround; and that people live through their practices and tacit knowledge so that the most powerful things are those that are effectively invisible in use. How can we make everyday computing conform better to these principles of human effectiveness?

Our approach: Activate the world. Provide hundreds of wireless computing devices per person per office, of all scales (from 1" displays to wall sized). We call it: "ubiquitous computing" — and it means the end of the personal computer. These things in the office, every place you are, are not workstations or PC's. You just grab anything that is nearby, and use it. Your information and state follows you; the hundreds of devices in your office or meeting room adjust to support you.

Pierre Wellner
What is the best way to interact with computers in our future offices? We could make workstations so powerful and so versatile that they integrate all our office activities in one place. Or, we could make virtual offices that could free us from all the constraints of the real world. Both of these approaches aim to eliminate the use of traditional tools such as paper and pencil, but my position is that we should do the opposite. Instead of *replacing* paper, pens, pencils, erasers, desks and lamps, we should keep them and use computers to augment their capabilities.

Let's take paper, for example. Screen-based documents have not (and will not) replace paper completely because, despite its limitations, paper is portable, cheap, universally accepted, high resolution, tactile and familiar. Like with many traditional tools, we hardly think about the skills we use to manipulate paper because they are embedded so deeply into our minds and bodies. A lot of effort goes into making electronic documents more like paper, but the approach I propose is to start with the paper, and augment it to behave more like electronic documents. With a projected display and video cameras that track fingertips and tools, we can create spaces in which everyday objects gain electronic properties without losing their familiar physical properties. I will briefly discuss work to date on a system that does this: the DigitalDesk. I will also show some future envisionment videos that illustrate more ideas for how we could augment (instead of replace) paper, pencils, erasers, desks and other parts of the traditional office.

References

Books and Articles

deBord, Guy, "Society of the Spectacle", *Black & Red* 1983

Polanyi, Michael, *The Tacit Dimension*, Doubleday & Company, Inc 1966.

Communications of the ACM, Special Issue on Augmented Environments, July 1993

Wellner, P., Interacting with Paper on the DigitalDesk. *Communications of the ACM*, July 1993.

Stifelman, Lisa J., Barry Arons, Chris Schmandt, and Eric A. Hulteen, VoiceNotes: A Speech Interface for a Hand-Held Voice Notetake, *Proceedings* of INTERCHI '93, ACM April 1993

Weiser, Mark, "The Computer for the 21st Century," *Scientific American*, Sept. 1991

Bajura, M., H. Fuchs, and R. Ohbuchi, Merging Virtual Objects with the Real World: Seeing Ultrasound Imagery within the Patient. *Computer Graphics* (Proc. SIGGRAPH '92) 26(2):203-210, July, 1992.

Feiner, S. and A. Shamash, Hybrid User Interfaces: Breeding Virtually Bigger Interfaces for Physically Smaller Computers. *Proc.* UIST '91 (ACM Symp. on User Interface Software and Technology), Hilton Head, SC, November 11-13, 1991, 9-17.

Feiner, S., B. MacIntyre, and D. Seligmann, Annotating the Real World with Knowledge-Based Graphics on a See-Through Head-Mounted Display, In *Proc.* Graphics Interface '92, pages 78-85. Vancouver, Canada, May 11-15, 1992.

Feiner, S., B. MacIntyre, and D. Seligmann, Knowledge-Based Augmented Reality, *Communications of the ACM*, 36(7), July 1993.

Magazines

Electronic House, published bi-monthly by Electronic House Inc.

Merging 3-D Graphics and Imaging - Applications and Issues

Chair
William R. Pickering, Silicon Graphics Computer Systems

Panelists:
Paul Douglas,Earthwatch Communications
Kevin Hussey, Jet Propulsion Laboratory
Michael Natkin, Industrial Light and Magic

There are many applications that require both image processing and 3D graphics techniques. The panelists present application specifics and demonstrations of how image processing and 3D graphics are used together, and why these applications cannot be solved without the use of both disciplines.

Panel Background
Computer graphics and image processing were once two very distinct disciplines, with different hardware, software, and users. Now they are becoming increasingly intertwined. Applications are being developed that use both 3D graphics and imaging techniques for a broad spectrum of uses, including realistic scene simulations, interesting visual effects, and improved analysis and understanding of complex information. What should graphics users learn from their imaging counterparts, and vice versa? This panel brings together people from both disciplines who have been developing a variety of applications that merge graphics and imaging technologies. They will illustrate both the benefits and current limitations.

Panel Goals and Issues
Many applications require the use of real-world data most often obtained from sensors in the form of images. To view the data in the three dimensional space that it was obtained from, 3D graphics techniques are often used. How is image data transformed into a 3D view? The solution has been approached from both the image processing and 3D graphics disciplines.

Image processing technology is used to take real-world image data and process it into an image that highlights desired features. The final image can then be used in conjunction with 3D graphics techniques to render a 3D scene.

How are images used in conjunction with 3D graphics techniques to create more realistic views? What image processing techniques are used to enhance the image before rendering? How can images be utilized as sources for 3D models? How can 3D graphics techniques be used to view the image data?

The panel will offer discussion and demonstrations of how image processing and 3D graphics are jointly used to create 3D scenes from 2D images. Questions from the attendees will be answered during the concluding portion of the session.

Kevin Hussey
Kevin will describe the image processing techniques necessary for texture map preparation in the context of a complex scientific visualization entitled "Monterey: The Bay". The visualization simulates a flight through the Monterey Bay environment (above and below the ocean surface) using seven different geophysical data sets. Data from a numerical model of ocean currents were interpolated into sufficient time intervals for a smooth animation. Sets of polygonal "ribbons" were created to visualize the current flow. These ribbons were then rendered, anti-aliased and composited into a perspective ray-cast scene comprised of 3 additional image data sets.

Paul Douglas
Paul will describe a weather graphics system that he has developed and uses to provide visuals for his nightly weather broadcasts. The system transforms flat, 2-D pixel imagery of clouds and radar data into 3-D geometry allowing the viewer to view weather from multiple perspectives. The system merges images, ingested from live weather feeds including geostationary weather satellites over the Pacific, Europe and North America., with a full GIS system containing data from all over the world, to create a 3-D model. Weather phenomena is supplemented with imagery from SPOT, Landsat or aerial photography for enhanced realism.

Michael Natkin
Successful special effects depend on the seamless integration of computer generated objects with live action environments on film. Two-dimensional image techniques are a critical component of Industrial Light and Magic's production process. Michael will describe the use of painting, digital compositing, color balancing and edge quality adjustment, wire removal, image warping (morphing), texture projection, and 3D painting in movies such as *Terminator 2* and *Jurassic Park.* He will also discuss the future of special effects image processing, design of software for special effects production, and system hardware and software performance requirements.

Nan-o-sex and Virtual Seduction

Co-chairs:
Joan I. Staveley, Artist/Educator
David Steiling, Ringling School of Art and Design

Panelists:
Paul Brown, Mississippi State University
Michael Heim, Philosopher/Author
Jill Hunt, Angel Studios
Chitra Shriram, Xaos Inc. and The Ohio State University

Goals and Principle Issues
Among the principal issues the panel should address are:
1) Is virtual space really interactive or merely voyeuristic?
2) What are some of the alternatives to classic male erotica that might unfold within the possibilities of virtual space?
3) Who will be constructing the space? Who will be the audience? Who will be the target market?
4) Who/what are the real subjects and objects in virtual eroticism? How are the subject and objects constructed?
5) Is virtual reality a challenge or an aid to the development of better understanding and relations between the genders?
6) Is Cartesian coordinate space "neutral" or is it, like language, male-constructed?

The purpose of the discourse is not to add to cross-gender recriminations or to arrive at satisfactory conclusions, but to start a discussion based on the inevitability that eroticism is one of the first uses to which virtual reality is being put.

Panel Background
This panel is organized to apply some of the techniques and positions of recent criticism to virtual reality and computer graphics. The panel is committed to doing this in a manner that would avoid much of the jargon that can make such discussions hard to follow. The panel also wishes to avoid moral judgments on the subject being discussed in favor of allowing a plurality of voices to be heard on this controversial topic. Among the positions that will inform the discussion are those of feminism, deconstruction, Marxism, and a number of other post-modern critical attitudes. Among the assumptions of the panel is that, like it or not, virtual reality eroticism is going to be with us, and is in fact avidly anticipated.

Paul Brown
Some have proposed that we can expect virtual realities that are as 'fine grained' as physical reality by the year 2000. At this point our whole appreciation of "the natural world" may well break down. Instead of a defined reality, with myriad imaginary worlds, we will have a continuum of potential realities where the 'real' reality will have lost any particular significance. Our model is the renaissance period when the planet Earth changed from the center of the universe to an undistinguished speck in an unremarkable galaxy in an inconceivably vast and growing universe. The sexual implications of this breakdown of reality are tantalizing. The likely result is a tight symbiosis which will include humankind and their machines—virtual sex will be existentially 'real sex' where mutual human activity will be only one manifestation, distinguished perhaps only by the amount of mess it makes. Researchers report that women prefer linguistic eroticism while men prefer visual (Iconic) forms. The new eroticism with its immersive, symbiotic qualities may well bridge the gender gap. A lot will depend on who controls it. If it is developed within the traditional male hierarchical power structure it could become yet another form of constraint; however, the networks that will enable this evolution have an intrinsic matriarchal hetrachical quality. The growing number of technical conferences which have responded to the need for critical discussion of human/gender oriented issues is evidence of a move towards a more humanistic mode of enquirey.

Michael Heim
Primitive VR has become a screen on which we project our fantasies, both collectively and individually. Soon, the entertainment industry will sell gender-based sex fantasies for virtually safe sex. But the hazards of teledildonics will trap sex in the same voyeuristic cage that imprisoned painting, the novel, and cinema. Computer-generated realities have sprung from a deep erotic drive to see, hear, touch, and be touched by a perfect, knowable world. The thrill of speed, the elan of flight, and the freedom of a liquid self have attracted humans for centuries. Plug this Eros into electronics, and you think you can have it both ways, keep a distance while at the same time put yourself on the line. Beyond gender voyeurism moves a freely active and self-aware body that revels in the flow of internal energy without genital fixation. VR could eventually become a tool for training and sharing this body of spiral energy. Rather than a corpse dangling in frustration while trying to 'have it both ways,' the old erotic body may learn to move and love in entirely new ways.

Jill Hunt
Currently VR, in terms of eroticism, is purely voyeuristic. In the future, VR gender-independent interactive spaces could be used a creative outlet to help bridge the gap between the sexes, particularly in the workplace, creating a more direct method of communication and information exchange. By developing gender-independent spaces we might also risk repressing gender awareness altogether; therefore we must also develop gender-dependent experiences and exchanges in order to further our awareness of ourselves and the opposite sex. There is room for more variety of expression than is currently predominant in our male dominated society, although it will take time to develop and educate people on the benefits of this exposure. Repression is imbedded in our society and directly related to gender discrimination and sexuality. The current and confused view of repression causes many problems that I feel can be work out in VR.

Chitra Shriram
Battles for power, money and ideology rage about the still largely anticipated phenomena of virtual reality. An ancient myth comes to mind. Gods and Demons had once grown so powerful that neither could gain and advantage of the other. Each side strove to prevail by drinking the nectar of immortality that flowed deep in the ocean depths, so Gods and demons together began to churn the ocean. Incredible gifts spewed forth and both nectar and poison rose to the surface of the water, inviting consumption. The god Shiva drank the poison, arresting the evil in his throat. He saved the world from destruction. The God Vishnu, disguised as an enchantress, lured

away the demons so that the Gods could create good with the life-giving nectar.

We barely know what poisons or nectars we court. The very notion of looking to ancient spiritual disciplines to inform us on how and what we could do with our latest scientific and technological feats is likely to sound ludicrous—tolerated only in the convenient name of 'multiculturalism.' But the very absurdity of the exercise is warranted by an urgent need to widen the base of value systems which nourish this new manifestation of our psychic and technological desire and energy.

References

Benedikt, Michael (ed). *Cyberspace: First Steps.* MA: The MIT Press, 1992.

Bodychello, *Heavenly Bodies*, Multimedia Publication 1993.

Bonobo Production, *Digital Dancing*, Multimedia Publication 1993.

Brand, Steward, *The Media Lab - Inventing the Future at MIT*, Viking/Penguin, 1987.

Bright, Susie. *Susie Bright's Sexual Reality: A Virtual Sex World Reader.* USA: Cleis Press, Inc., 1992.

Bright, Susie; Blank, Joani. *Herotica: A Collection of Women's Erotic Fiction.* New York: Penguin Group, 1992

Brown, Paul. "Metamedia and Cyberspace - Advanced Computers and the Future of Art," in Hayward, Philip (ed.), *Culture Technology and Creativity in the Late Twentieth Century.* Arts Council of Great Britain and John Lebby Press, London, 1990. See also Paul Brown, "Beyond Art," in Clifford Pickover (ed.), *Visions of the Future*, To appear 1992.

Brown, Paul. "Communion and Cargo Cults," Proceedings of the Second International Symposium on Electronic Arts - SISEA, Groningen, The Netherlands 1990.

Brown, Paul. "The Convergence of Reality and Illusion," Proceedings Hi-Vision 92, Tokyo, 1992.

Brown, Paul. "Reality versus Illusion," Visual Proceedings SIGGRAPH 92, Chicago, July 1992.

Brown, Paul. "Symbiosis, Semiosis and Communication Technology - The Ethics and Aesthetics of Computer Human Interaction." To appear.

Dorsey, Candas Jane, *Machine Sex and Other Stories*, Women's Press, London 1988.

Faludi, Susan. *Backlash: The Undeclared War on American Women.* New York: An Anchor Book, Doubleday, 1991.

Foster, Hal (ed). *The Anti-Aesthetic: Essays on Postmodern Culture.* WA: Bay Press, 1983.

Gibson, William. The Cyperpunk Trilogy - *Neuromancer*, Gollancz London 1984; *Count Zero*, Gollancz London 1986; *Mona Lisa Overdrive*, Gollancz London 1988.

Handhardt, John (ed). *Video Culture: A Critical Investigation.* New York: Visual Studies Workshop, 1986.

Heim, Michael. *The Metaphysics of Virtual Reality*, with a foreword by Myron Krueger, Oxford University Press, 1993.

Heim, Michael. *Electric Language: A Philosophical Study of Word Processing*, Yale Univesity Press, 1989 Yale paperback, c1987.

Heim, Michael. "The Computer as Component: Heidegger and McLuhan," *Philosophy and Literature*, Johns Hopkins University Press, October 1992.

Heim, Michael. "Cybersage Does Tai Chi," in the anthology *Falling in Love with Wisdom: On Becoming a Philosopher*, David Karnos and Robert Shoemaker (eds.), Oxford University Press, 1993.

Heim, Michael. "The Erotic Ontology of Cyberspace," in *Cyberspace: First Steps*, edited by Michael Benedikt, MIT Press 1991. An early version of the paper was a talk at the First Conference on Cyberspace held at the University of Texas in Austin, May 4-5, 1990.

Heim, Michael. "The Metaphysics of Virtual Reality," *Multimedia Review*, Issue Three ("New Paradigms"), published by Meckler Publishing. This essay also appears in the Meckler reprint of Issue Two ("Virtual Reality") and was the basis for a talk on the program "Virtual Reality: Theory, Practice, and Promise" in San Francisco, December 10, 1990.

Hershman, Lynn. *Deep Contact - the Incomplete Sexual Fantasy Disk*, CD-ROM 1990.

Laurel, Brenda. "On Finger Flying & Other Faulty Notions," in Linda Jacobson (ed.), *CyberArts*, Miller Freeman 1992.

Laurel, Brenda. *Computers as Theatre.* Addison-Wesley Publishing Company, 1991.

Mulvey, Laura. "Visual Pleasure and Narrative Cinema," in the anthology *The Sexual Subject: A Screen Reader in Sexuality*, edited by *Screen*, Routledge 1992.

Mondo 2000, California: Published by Fun City Megamedia.

Nelson, Ted, *Computer Lib/Dream Machines*, republished in a revised and updated edition by Tempus/Microsoft in 1987.

Reactor Software, *Virtual Valerie*, Multimedia Publication, 1993.

Rheingold, Howard. *Virtual Reality.* New York: Simon & Schuster Inc., Copyright 1991 Howard Rheingold.

Rucker, Rudy, "R U Sirius and Queen Mu, A Users Guide to the New Edge," *Mondo 2000*, 1992.

Singer, Linda. *Erotic Welfare: Sexual Theory and Politics in the Age of Epidemic*, Routledge, Ny, London 1993.

Stefanac, Susanne. "Sex and the New Media," *New Media Magazine*, Vol. 3 No. 4, April 1993, pp. 38-45.

Wired, California: Published by Louis Rossetto.