correspondence algorithms) is that uniqueness of matches is only enforced for one image (the *reference image*), while points in the other image might match multiple points, unless cross-checking and subsequent hole filling is used (Fua 1993; Hirschmüller and Scharstein 2009).

11.4.1 Sub-pixel estimation and uncertainty

Most stereo correspondence algorithms compute a set of disparity estimates in some discretized space, e.g., for integer disparities (exceptions include continuous optimization techniques such as optical flow (Bergen, Anandan, Hanna *et al.* 1992) or splines (Szeliski and Coughlan 1997)). For applications such as robot navigation or people tracking, these may be perfectly adequate. However for image-based rendering, such quantized maps lead to very unappealing view synthesis results, i.e., the scene appears to be made up of many thin shearing layers. To remedy this situation, many algorithms apply a sub-pixel refinement stage after the initial discrete correspondence stage. (An alternative is to simply start with more discrete disparity levels (Szeliski and Scharstein 2004).)

Sub-pixel disparity estimates can be computed in a variety of ways, including iterative gradient descent and fitting a curve to the matching costs at discrete disparity levels (Ryan, Gray, and Hunt 1980; Lucas and Kanade 1981; Tian and Huhns 1986; Matthies, Kanade, and Szeliski 1989; Kanade and Okutomi 1994). This provides an easy way to increase the resolution of a stereo algorithm with little additional computation. However, to work well, the intensities being matched must vary smoothly, and the regions over which these estimates are computed must be on the same (correct) surface.

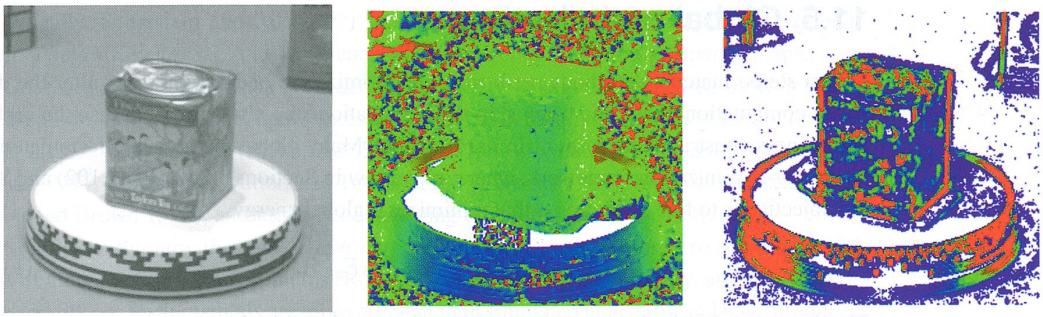
Recently, some questions have been raised about the advisability of fitting correlation curves to integer-sampled matching costs (Shimizu and Okutomi 2001). This situation may even be worse when sampling-insensitive dissimilarity measures are used (Birchfield and Tomasi 1998). These issues are explored in more depth by Szeliski and Scharstein (2004).

Besides sub-pixel computations, there are other ways of post-processing the computed disparities. Occluded areas can be detected using cross-checking, i.e., comparing left-to-right and right-to-left disparity maps (Fua 1993). A median filter can be applied to clean up spurious mismatches, and holes due to occlusion can be filled by surface fitting or by distributing neighboring disparity estimates (Birchfield and Tomasi 1999; Scharstein 1999; Hirschmüller and Scharstein 2009).

Another kind of post-processing, which can be useful in later processing stages, is to associate *confidences* with per-pixel depth estimates (Figure 11.10), which can be done by looking at the curvature of the correlation surface, i.e., how strong the minimum in the DSI image is at the winning disparity. Matthies, Kanade, and Szeliski (1989) show that under the assumption of small noise, photometrically calibrated images, and densely sampled disparities, the variance of a local depth estimate can be estimated as

$$Var(d) = \frac{\sigma_I^2}{a}, \tag{11.7}$$

where a is the curvature of the DSI as a function of d, which can be measured using a local parabolic fit or by squaring all the horizontal gradients in the window, and σ_I^2 is the variance of the image noise, which can be estimated from the minimum SSD score. (See also Section 6.1.4, (8.44), and Appendix B.6.)



(a) (b) (c)

Figure 11.10 Uncertainty in stereo depth estimation (Szeliski 1991b): (a) input image; (b) estimated depth map (blue is closer); (c) estimated confidence(red is higher). As you can see, more textured areas have higher confidence.

11.4.2 *Application*: Stereo-based head tracking

A common application of real-time stereo algorithms is for tracking the position of a user interacting with a computer or game system. The use of stereo can dramatically improve the reliability of such a system compared to trying to use monocular color and intensity information (Darrell, Gordon, Harville *et al.* 2000). Once recovered, this information can be used in a variety of applications, including controlling a virtual environment or game, correcting the apparent gaze during video conferencing, and background replacement. We discuss the first two applications below and defer the discussion of background replacement to Section 11.5.3.

The use of head tracking to control a user's virtual viewpoint while viewing a 3D object or environment on a computer monitor is sometimes called *fish tank virtual reality*, since the user is observing a 3D world as if it were contained inside a fish tank (Ware, Arthur, and Booth 1993). Early versions of these systems used mechanical head tracking devices and stereo glasses. Today, such systems can be controlled using stereo-based head tracking and stereo glasses can be replaced with autostereoscopic displays. Head tracking can also be used to construct a "virtual mirror", where the user's head can be modified in real-time using a variety of visual effects (Darrell, Baker, Crow *et al.* 1997).

Another application of stereo head tracking and 3D reconstruction is in gaze correction (Ott, Lewis, and Cox 1993). When a user participates in a desktop video-conference or video chat, the camera is usually placed on top of the monitor. Since the person is gazing at a window somewhere on the screen, it appears as if they are looking down and away from the other participants, instead of straight at them. Replacing the single camera with two or more cameras enables a virtual view to be constructed right at the position where they are looking resulting in virtual eye contact. Real-time stereo matching is used to construct an accurate 3D head model and view interpolation (Section 13.1) is used to synthesize the novel in-between view (Criminisi, Shotton, Blake *et al.* 2003).

11.5 Global optimization

Global stereo matching methods perform some optimization or iteration steps after the disparity computation phase and often skip the aggregation step altogether, because the global smoothness constraints perform a similar function. Many global methods are formulated in an energy-minimization framework, where, as we saw in Sections 3.7 (3.100–3.102) and 8.4, the objective is to find a solution d that minimizes a global energy,

$$E(d) = E_d(d) + \lambda E_s(d). \tag{11.8}$$

The data term, $E_d(d)$, measures how well the disparity function d agrees with the input image pair. Using our previously defined disparity space image, we define this energy as

$$E_d(d) = \sum_{(x,y)} C(x, y, d(x, y)), \tag{11.9}$$

where C is the (initial or aggregated) matching cost DSI.

The smoothness term $E_s(d)$ encodes the smoothness assumptions made by the algorithm. To make the optimization computationally tractable, the smoothness term is often restricted to measuring only the differences between neighboring pixels' disparities,

$$E_s(d) = \sum_{(x,y)} \rho(d(x, y) - d(x + 1, y)) + \rho(d(x, y) - d(x, y + 1)), \tag{11.10}$$

where ρ is some monotonically increasing function of disparity difference. It is also possible to use larger neighborhoods, such as $\mathcal{N}_8$, which can lead to better boundaries (Boykov and Kolmogorov 2003), or to use second-order smoothness terms (Woodford, Reid, Torr *et al.* 2008), but such terms require more complex optimization techniques. An alternative to smoothness functionals is to use a lower-dimensional representation such as splines (Szeliski and Coughlan 1997).

In standard regularization (Section 3.7.1), ρ is a quadratic function, which makes d smooth everywhere and may lead to poor results at object boundaries. Energy functions that do not have this problem are called *discontinuity-preserving* and are based on robust ρ functions (Terzopoulos 1986b; Black and Rangarajan 1996). The seminal paper by Geman and Geman (1984) gave a Bayesian interpretation of these kinds of energy functions and proposed a discontinuity-preserving energy function based on Markov random fields (MRFs) and additional *line processes*, which are additional binary variables that control whether smoothness penalties are enforced or not. Black and Rangarajan (1996) show how independent line process variables can be replaced by robust pairwise disparity terms.

The terms in E_s can also be made to depend on the intensity differences, e.g.,

$$\rho_d(d(x, y) - d(x + 1, y)) \cdot \rho_I(\|I(x, y) - I(x + 1, y)\|), \tag{11.11}$$

where ρ_I is some monotonically decreasing function of intensity differences that lowers smoothness costs at high-intensity gradients. This idea (Gamble and Poggio 1987; Fua 1993; Bobick and Intille 1999; Boykov, Veksler, and Zabih 2001) encourages disparity discontinuities to coincide with intensity or color edges and appears to account for some of the good performance of global optimization approaches. While most researchers set these functions

heuristically, Scharstein and Pal (2007) show how the free parameters in such *conditional random fields* (Section 3.7.2, (3.118)) can be learned from ground truth disparity maps.

Once the global energy has been defined, a variety of algorithms can be used to find a (local) minimum. Traditional approaches associated with regularization and Markov random fields include continuation (Blake and Zisserman 1987), simulated annealing (Geman and Geman 1984; Marroquin, Mitter, and Poggio 1987; Barnard 1989), highest confidence first (Chou and Brown 1990), and mean-field annealing (Geiger and Girosi 1991).

More recently, *max-flow* and *graph cut* methods have been proposed to solve a special class of global optimization problems (Roy and Cox 1998; Boykov, Veksler, and Zabih 2001; Ishikawa 2003). Such methods are more efficient than simulated annealing and have produced good results, as have techniques based on loopy belief propagation (Sun, Zheng, and Shum 2003; Tappen and Freeman 2003). Appendix B.5 and a recent survey paper on MRF inference (Szeliski, Zabih, Scharstein *et al.* 2008) discuss and compare such techniques in more detail.

While global optimization techniques currently produce the best stereo matching results, there are some alternative approaches worth studying.

Cooperative algorithms. Cooperative algorithms, inspired by computational models of human stereo vision, were among the earliest methods proposed for disparity computation (Dev 1974; Marr and Poggio 1976; Marroquin 1983; Szeliski and Hinton 1985; Zitnick and Kanade 2000). Such algorithms iteratively update disparity estimates using non-linear operations that result in an overall behavior similar to global optimization algorithms. In fact, for some of these algorithms, it is possible to explicitly state a global function that is being minimized (Scharstein and Szeliski 1998).

Coarse-to-fine and incremental warping. Most of today's best algorithms first enumerate all possible matches at all possible disparities and then select the best set of matches in some way. Faster approaches can sometimes be obtained using methods inspired by classic (infinitesimal) optical flow computation. Here, images are successively warped and disparity estimates incrementally updated until a satisfactory registration is achieved. These techniques are most often implemented within a coarse-to-fine hierarchical refinement framework (Quam 1984; Bergen, Anandan, Hanna *et al.* 1992; Barron, Fleet, and Beauchemin 1994; Szeliski and Coughlan 1997).

11.5.1 Dynamic programming

A different class of global optimization algorithm is based on *dynamic programming*. While the 2D optimization of Equation (11.8) can be shown to be NP-hard for common classes of smoothness functions (Veksler 1999), dynamic programming can find the global minimum for independent scanlines in polynomial time. Dynamic programming was first used for stereo vision in sparse, edge-based methods (Baker and Binford 1981; Ohta and Kanade 1985). More recent approaches have focused on the dense (intensity-based) scanline matching problem (Belhumeur 1996; Geiger, Ladendorf, and Yuille 1992; Cox, Hingorani, Rao *et al.* 1996; Bobick and Intille 1999; Birchfield and Tomasi 1999). These approaches work by computing the minimum-cost path through the matrix of all pairwise matching costs between two corresponding scanlines, i.e., through a horizontal slice of the DSI. Partial occlusion is

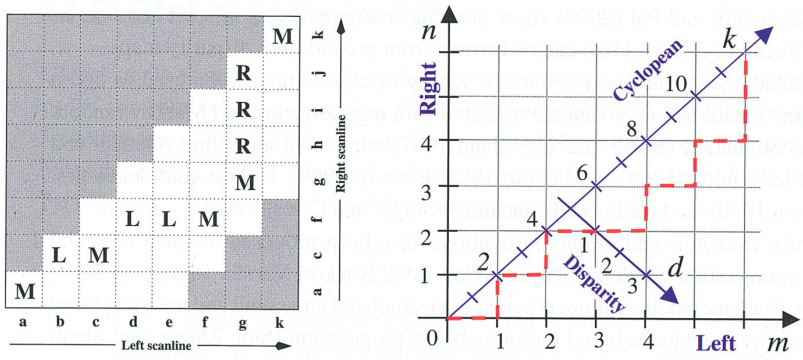


Figure 11.11 Stereo matching using dynamic programming, as illustrated by (a) Scharstein and Szeliski (2002) © 2002 Springer and (b) Kolmogorov, Criminisi, Blake *et al.* (2006). © 2006 IEEE. For each pair of corresponding scanlines, a minimizing path through the matrix of all pairwise matching costs (DSI) is selected. Lowercase letters (a–k) symbolize the intensities along each scanline. Uppercase letters represent the selected path through the matrix. Matches are indicated by M, while partially occluded points (which have a fixed cost) are indicated by L or R, corresponding to points only visible in the left or right images, respectively. Usually, only a limited disparity range is considered (0–4 in the figure, indicated by the non-shaded squares). The representation in (a) allows for diagonal moves while the representation in (b) does not. Note that these diagrams, which use the *Cyclopean* representation of depth, i.e., depth relative to a camera between the two input cameras, show an "unskewed" x-d slice through the DSI.

handled explicitly by assigning a group of pixels in one image to a single pixel in the other image. Figure 11.11 schematically shows how DP works, while Figure 11.5f shows a real DSI slice over which the DP is applied.

To implement dynamic programming for a scanline y, each entry (state) in a 2D cost matrix $D(m,n)$ is computed by combining its DSI value

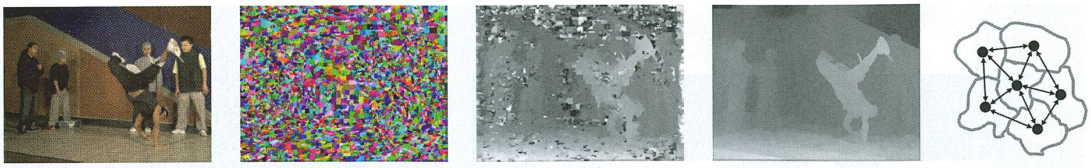
$$C'(m,n) = C(m+n, m-n, y) \tag{11.12}$$

with one of its predecessor cost values. Using the representation shown in Figure 11.11a, which allows for "diagonal" moves, the aggregated match costs can be recursively computed as

$$\begin{aligned} D(m,n,M) &= \min(D(m-1,n-1,M), D(m-1,n,L), D(m-1,n-1,R)) \\ &\quad + C'(m,n) \\ D(m,n,L) &= \min(D(m-1,n-1,M), D(m-1,n,L)) + O \\ D(m,n,R) &= \min(D(m,n-1,M), D(m,n-1,R)) + O, \end{aligned} \tag{11.13}$$

where O is a per-pixel occlusion cost. The aggregation rules corresponding to Figure 11.11b are given by Kolmogorov, Criminisi, Blake *et al.* (2006), who also use a two-state foreground–background model for bi-layer segmentation.

Problems with dynamic programming stereo include the selection of the right cost for occluded pixels and the difficulty of enforcing inter-scanline consistency, although several



(a) (b) (c) (d) (e)

Figure 11.12 Segmentation-based stereo matching (Zitnick, Kang, Uyttendaele *et al.* 2004) © 2004 ACM: (a) input color image; (b) color-based segmentation; (c) initial disparity estimates; (d) final piecewise-smoothed disparities; (e) MRF neighborhood defined over the segments in the disparity space distribution (Zitnick and Kang 2007) © 2007 Springer.

methods propose ways of addressing the latter (Ohta and Kanade 1985; Belhumeur 1996; Cox, Hingorani, Rao *et al.* 1996; Bobick and Intille 1999; Birchfield and Tomasi 1999; Kolmogorov, Criminisi, Blake *et al.* 2006). Another problem is that the dynamic programming approach requires enforcing the *monotonicity* or *ordering constraint* (Yuille and Poggio 1984). This constraint requires that the relative ordering of pixels on a scanline remain the same between the two views, which may not be the case in scenes containing narrow foreground objects.

An alternative to traditional dynamic programming, introduced by Scharstein and Szeliski (2002), is to neglect the vertical smoothness constraints in (11.10) and simply optimize independent scanlines in the global energy function (11.8), which can easily be done using a recursive algorithm,

$$D(x, y, d) = C(x, y, d) + \min_{d'} \{D(x - 1, y, d') + \rho_d(d - d')\}. \tag{11.14}$$

The advantage of this *scanline optimization* algorithm is that it computes the same representation and minimizes a reduced version of the same energy function as the full 2D energy function (11.8). Unfortunately, it still suffers from the same streaking artifacts as dynamic programming.

A much better approach is to evaluate the cumulative cost function (11.14) from multiple directions, e.g, from the eight cardinal directions, N, E, W, S, NE, SE, SW, NW (Hirschmüller 2008). The resulting *semi-global* optimization performs quite well and is extremely efficient to implement.

Even though dynamic programming and scanline optimization algorithms do not generally produce *the* most accurate stereo reconstructions, when combined with sophisticated aggregation strategies, they can produce very fast and high-quality results.

11.5.2 Segmentation-based techniques

While most stereo matching algorithms perform their computations on a per-pixel basis, some of the more recent techniques first segment the images into regions and then try to label each region with a disparity.

For example, Tao, Sawhney, and Kumar (2001) segment the reference image, estimate per-pixel disparities using a local technique, and then do local plane fits inside each segment

(a) (b)

Figure 11.13 Stereo matching with adaptive over-segmentation and matting (Taguchi, Wilburn, and Zitnick 2008) © 2008 IEEE: (a) segment boundaries are refined during the optimization, leading to more accurate results (e.g., the thin green leaf in the bottom row); (b) alpha mattes are extracted at segment boundaries, which leads to visually better compositing results (middle column).

before applying smoothness constraints between neighboring segments. Zitnick, Kang, Uyttendaele *et al.* (2004) and Zitnick and Kang (2007) use over-segmentation to mitigate initial bad segmentations. After a set of initial cost values for each segment has been stored into a *disparity space distribution* (DSD), iterative relaxation (or loopy belief propagation, in the more recent work of Zitnick and Kang (2007)) is used to adjust the disparity estimates for each segment, as shown in Figure 11.12. Taguchi, Wilburn, and Zitnick (2008) refine the segment shapes as part of the optimization process, which leads to much improved results, as shown in Figure 11.13.

Even more accurate results are obtained by Klaus, Sormann, and Karner (2006), who first segment the reference image using mean shift, run a small (3×3) SAD plus gradient SAD (weighted by cross-checking) to get initial disparity estimates, fit local planes, re-fit with global planes, and then run a final MRF on plane assignments with loopy belief propagation. When the algorithm was first introduced in 2006, it was the top ranked algorithm on the evaluation site at http://vision.middlebury.edu/stereo; in early 2010, it still had the top rank on the new evaluation datasets.

The highest ranked algorithm, by Wang and Zheng (2008), follows a similar approach of segmenting the image, doing local plane fits, and then performing cooperative optimization of neighboring plane fit parameters. Another highly ranked algorithm, by Yang, Wang, Yang *et al.* (2009), uses the color correlation approach of Yoon and Kweon (2006) and hierarchical belief propagation to obtain an initial set of disparity estimates. After left–right consistency checking to detect occluded pixels, the data terms for low-confidence and occluded pixels are recomputed using segmentation-based plane fits and one or more rounds of hierarchical belief propagation are used to obtain the final disparity estimates.

Another important ability of segmentation-based stereo algorithms, which they share with algorithms that use explicit layers (Baker, Szeliski, and Anandan 1998; Szeliski and Golland 1999) or boundary extraction (Hasinoff, Kang, and Szeliski 2006), is the ability to extract fractional pixel alpha mattes at depth discontinuities (Bleyer, Gelautz, Rother *et al.* 2009). This ability is crucial when attempting to create virtual view interpolation without clinging boundary or tearing artifacts (Zitnick, Kang, Uyttendaele *et al.* 2004) and also to seamlessly insert virtual objects (Taguchi, Wilburn, and Zitnick 2008), as shown in Figure 11.13b.

Figure 11.14 Background replacement using z-keying with a bi-layer segmentation algorithm (Kolmogorov, Criminisi, Blake *et al.* 2006) © 2006 IEEE.

Since new stereo matching algorithms continue to be introduced every year, it is a good idea to periodically check the Middlebury evaluation site at http://vision.middlebury.edu/stereo for a listing of the most recent algorithms to be evaluated.

11.5.3 *Application*: Z-keying and background replacement

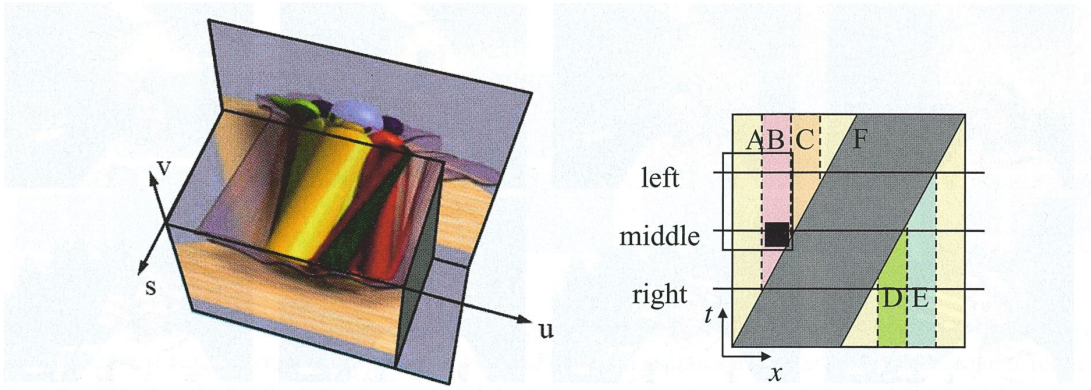
Another application of real-time stereo matching is *z-keying*, which is the process of segmenting a foreground actor from the background using depth information, usually for the purpose of replacing the background with some computer-generated imagery, as shown in Figure 11.2g.

Originally, z-keying systems required expensive custom-built hardware to produce the desired depth maps in real time and were, therefore, restricted to broadcast studio applications (Kanade, Yoshida, Oda *et al.* 1996; Iddan and Yahav 2001). Off-line systems were also developed for estimating 3D multi-viewpoint geometry from video streams (Section 13.5.4) (Kanade, Rander, and Narayanan 1997; Carranza, Theobalt, Magnor *et al.* 2003; Zitnick, Kang, Uyttendaele *et al.* 2004; Vedula, Baker, and Kanade 2005). Recent advances in highly accurate real-time stereo matching, however, now make it possible to perform z-keying on regular PCs, enabling desktop videoconferencing applications such as those shown in Figure 11.14 (Kolmogorov, Criminisi, Blake *et al.* 2006).

11.6 Multi-view stereo

While matching pairs of images is a useful way of obtaining depth information, matching more images can lead to even better results. In this section, we review not only techniques for creating complete 3D object models, but also simpler techniques for improving the quality of depth maps using multiple source images.

As we saw in our discussion of plane sweep (Section 11.1.2), it is possible to resample all neighboring k images at each disparity hypothesis d into a generalized disparity space



(a) (b)

Figure 11.15 Epipolar plane image (EPI) (Gortler, Grzeszczuk, Szeliski *et al.* 1996) © 1996 ACM and a schematic EPI (Kang, Szeliski, and Chai 2001) © 2001 IEEE. (a) The Lumigraph (light field) (Section 13.3) is the 4D space of all light rays passing through a volume of space. Taking a 2D slice results in all of the light rays embedded in a plane and is equivalent to a scanline taken from a stacked EPI volume. Objects at different depths move sideways with velocities (slopes) proportional to their inverse depth. Occlusion (and translucency) effects can easily be seen in this representation. (b) The EPI corresponding to Figure 11.16 showing the three images (middle, left, and right) as slices through the EPI volume. The spatially and temporally shifted window around the black pixel is indicated by the rectangle, showing the right image is not being used in matching.

volume $\tilde{I}(x, y, d, k)$. The simplest way to take advantage of these additional images is to sum up their differences from the reference image I_r as in (11.4),

$$C(x, y, d) = \sum_k \rho(\tilde{I}(x, y, d, k) - I_r(x, y)). \tag{11.15}$$

This is the basis of the well-known sum of summed-squared-difference (SSSD) and SSAD approaches (Okutomi and Kanade 1993; Kang, Webb, Zitnick *et al.* 1995), which can be extended to reason about likely patterns of occlusion (Nakamura, Matsuura, Satoh *et al.* 1996). More recent work by Gallup, Frahm, Mordohai *et al.* (2008) show how to adapt the baselines used to the expected depth in order to get the best tradeoff between geometric accuracy (wide baseline) and robustness to occlusion (narrow baseline). Alternative multi-view cost metrics include measures such as synthetic focus sharpness and the entropy of the pixel color distribution (Vaish, Szeliski, Zitnick *et al.* 2006).

A useful way to visualize the multi-frame stereo estimation problem is to examine the *epipolar plane image* (EPI) formed by stacking corresponding scanlines from all the images, as shown in Figures 8.13c and 11.15 (Bolles, Baker, and Marimont 1987; Baker and Bolles 1989; Baker 1989). As you can see in Figure 11.15, as a camera translates horizontally (in a standard horizontally rectified geometry), objects at different depths move sideways at a rate inversely proportional to their depth (11.1).[6] Foreground objects occlude background objects,

[6] The four-dimensional generalization of the EPI is the *light field*, which we study in Section 13.3. In principle, there is enough information in a light field to recover both the shape and the BRDF of objects (Soatto, Yezzi, and Jin 2003), although relatively little progress has been made to date on this topic.

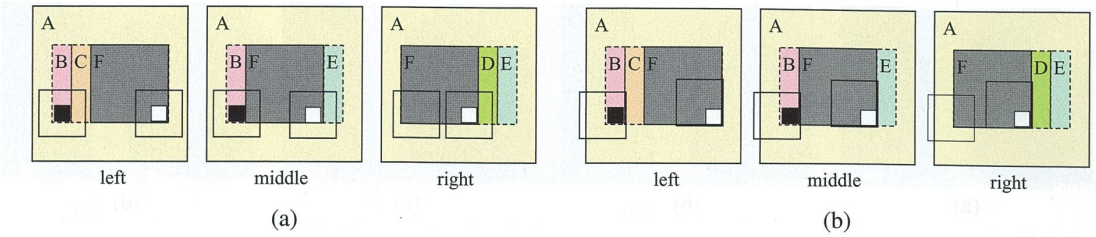


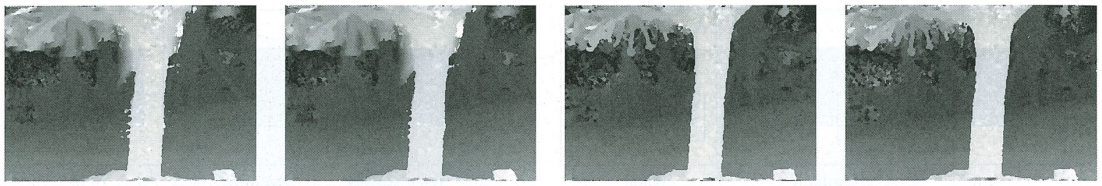
Figure 11.16 Spatio-temporally shiftable windows (Kang, Szeliski, and Chai 2001) © 2001 IEEE: A simple three-image sequence (the middle image is the reference image), which has a moving frontal gray square (marked F) and a stationary background. Regions B, C, D, and E are partially occluded. (a) A regular SSD algorithm will make mistakes when matching pixels in these regions (e.g. the window centered on the black pixel in region B) and in windows straddling depth discontinuities (the window centered on the white pixel in region F). (b) Shiftable windows help mitigate the problems in partially occluded regions and near depth discontinuities. The shifted window centered on the white pixel in region F matches correctly in all frames. The shifted window centered on the black pixel in region B matches correctly in the left image, but requires temporal selection to disable matching the right image. Figure 11.15b shows an EPI corresponding to this sequence and describes in more detail how temporal selection works.

which can be seen as *EPI-strips* (Criminisi, Kang, Swaminathan *et al.* 2005) occluding other strips in the EPI. If we are given a dense enough set of images, we can find such strips and reason about their relationships in order to both reconstruct the 3D scene and make inferences about translucent objects (Tsin, Kang, and Szeliski 2006) and specular reflections (Swaminathan, Kang, Szeliski *et al.* 2002; Criminisi, Kang, Swaminathan *et al.* 2005). Alternatively, we can treat the series of images as a set of sequential observations and merge them using Kalman filtering (Matthies, Kanade, and Szeliski 1989) or maximum likelihood inference (Cox 1994).

When fewer images are available, it becomes necessary to fall back on aggregation techniques such as sliding windows or global optimization. With additional input images, however, the likelihood of occlusions increases. It is therefore prudent to adjust not only the best window locations using a shiftable window approach, as shown in Figure 11.16a, but also to optionally select a subset of neighboring frames in order to discount those images where the region of interest is occluded, as shown in Figure 11.16b (Kang, Szeliski, and Chai 2001). Figure11.15b shows how such spatio-temporal selection or shifting of windows corresponds to selecting the most likely un-occluded volumetric region in the epipolar plane image volume.

The results of applying these techniques to the multi-frame *flower garden* image sequence are shown in Figure 11.17, which compares the results of using regular (non-shifted) SSSD with spatially shifted windows and full spatio-temporal window selection. (The task of applying stereo to a rigid scene filmed with a moving camera is sometimes called *motion stereo*). Similar improvements from using spatio-temporal selection are reported by (Kang and Szeliski 2004) and are evident even when local measurements are combined with global optimization.

While computing a depth map from multiple inputs outperforms pairwise stereo matching, even more dramatic improvements can be obtained by estimating multiple depth maps



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

Figure 11.17 Local (5×5 window-based) matching results (Kang, Szeliski, and Chai 2001) © 2001 IEEE: (a) window that is not spatially perturbed (centered); (b) spatially perturbed window; (c) using the best five of 10 neighboring frames; (d) using the better half sequence. Notice how the results near the tree trunk are improved using temporal selection.

simultaneously (Szeliski 1999; Kang and Szeliski 2004). The existence of multiple depth maps enables more accurate reasoning about occlusions, as regions which are occluded in one image may be visible (and matchable) in others. The multi-view reconstruction problem can be formulated as the simultaneous estimation of depth maps at key frames (Figure 8.13c) while maximizing not only photoconsistency and piecewise disparity smoothness but also the consistency between disparity estimates at different frames. While Szeliski (1999) and Kang and Szeliski (2004) use soft (penalty-based) constraints to encourage multiple disparity maps to be consistent, Kolmogorov and Zabih (2002) show how such consistency measures can be encoded as hard constraints, which guarantee that the multiple depth maps are not only similar but actually identical in overlapping regions. Newer algorithms that simultaneously estimate multiple disparity maps include papers by Maitre, Shinagawa, and Do (2008) and Zhang, Jia, Wong *et al.* (2008).

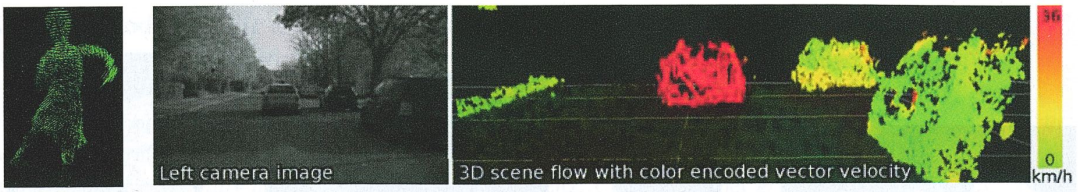
A closely related topic to multi-frame stereo estimation is *scene flow*, in which multiple cameras are used to capture a dynamic scene. The task is then to simultaneously recover the 3D shape of the object at every instant in time and to estimate the full 3D motion of every surface point between frames. Representative papers in this area include those by Vedula, Baker, Rander *et al.* (2005), Zhang and Kambhamettu (2003), Pons, Keriven, and Faugeras (2007), Huguet and Devernay (2007), and Wedel, Rabe, Vaudrey *et al.* (2008). Figure 11.18a shows an image of the 3D scene flow for the tango dancer shown in Figure 11.2h–j, while Figure 11.18b shows 3D scene flows captured from a moving vehicle for the purpose of obstacle avoidance. In addition to supporting mensuration and safety applications, scene flow can be used to support both spatial and temporal view interpolation (Section 13.5.4), as demonstrated by Vedula, Baker, and Kanade (2005).

11.6.1 Volumetric and 3D surface reconstruction

According to Seitz, Curless, Diebel *et al.* (2006):

> *The goal of multi-view stereo is to reconstruct a complete 3D object model from a collection of images taken from known camera viewpoints.*

The most challenging but potentially most useful variant of multi-view stereo reconstruction is to create globally consistent 3D models. This topic has a long history in computer vision, starting with surface mesh reconstruction techniques such as the one developed by



(a) (b)

Figure 11.18 Three-dimensional scene flow: (a) computed from a multi-camera dome surrounding the dancer shown in Figure 11.2h–j (Vedula, Baker, Rander *et al.* 2005) © 2005 IEEE; (b) computed from stereo cameras mounted on a moving vehicle (Wedel, Rabe, Vaudrey *et al.* 2008) © 2008 Springer.

Fua and Leclerc (1995) (Figure 11.19a). A variety of approaches and representations have been used to solve this problem, including 3D voxel representations (Seitz and Dyer 1999; Szeliski and Golland 1999; De Bonet and Viola 1999; Kutulakos and Seitz 2000; Eisert, Steinbach, and Girod 2000; Slabaugh, Culbertson, Slabaugh *et al.* 2004; Sinha and Pollefeys 2005; Vogiatzis, Hernandez, Torr *et al.* 2007; Hiep, Keriven, Pons *et al.* 2009), level sets (Faugeras and Keriven 1998; Pons, Keriven, and Faugeras 2007), polygonal meshes (Fua and Leclerc 1995; Narayanan, Rander, and Kanade 1998; Hernandez and Schmitt 2004; Furukawa and Ponce 2009), and multiple depth maps (Kolmogorov and Zabih 2002). Figure 11.19 shows representative examples of 3D object models reconstructed using some of these techniques.

In order to organize and compare all these techniques, Seitz, Curless, Diebel *et al.* (2006) developed a six-point taxonomy that can help classify algorithms according to the *scene representation*, *photoconsistency measure*, *visibility model*, *shape priors*, *reconstruction algorithm*, and *initialization requirements* they use. Below, we summarize some of these choices and list a few representative papers. For more details, please consult the full survey paper (Seitz, Curless, Diebel *et al.* 2006) and the evaluation Web site, http://vision.middlebury.edu/mview/, which contains pointers to even more recent papers and results.

Scene representation. One of the more popular 3D representations is a uniform grid of 3D voxels,[7] which can be reconstructed using a variety of carving (Seitz and Dyer 1999; Kutulakos and Seitz 2000) or optimization (Sinha and Pollefeys 2005; Vogiatzis, Hernandez, Torr *et al.* 2007; Hiep, Keriven, Pons *et al.* 2009) techniques. Level set techniques (Section 5.1.4) also operate on a uniform grid but, instead of representing a binary occupancy map, they represent the signed distance to the surface (Faugeras and Keriven 1998; Pons, Keriven, and Faugeras 2007), which can encode a finer level of detail. Polygonal meshes are another popular representation (Fua and Leclerc 1995; Narayanan, Rander, and Kanade 1998; Isidoro and Sclaroff 2003; Hernandez and Schmitt 2004; Furukawa and Ponce 2009; Hiep, Keriven, Pons *et al.* 2009). Meshes are the standard representation used in computer graphics and also readily support the computation of visibility and occlusions. Finally, as we discussed in the previous section, multiple depth maps can also be used (Szeliski 1999; Kolmogorov and Zabih 2002; Kang and Szeliski 2004). Many algorithms also use more than a single representation, e.g., they may start by computing multiple depth maps and then merge

[7] For outdoor scenes that go to infinity, a non-uniform gridding of space may be preferable (Slabaugh, Culbertson, Slabaugh *et al.* 2004).

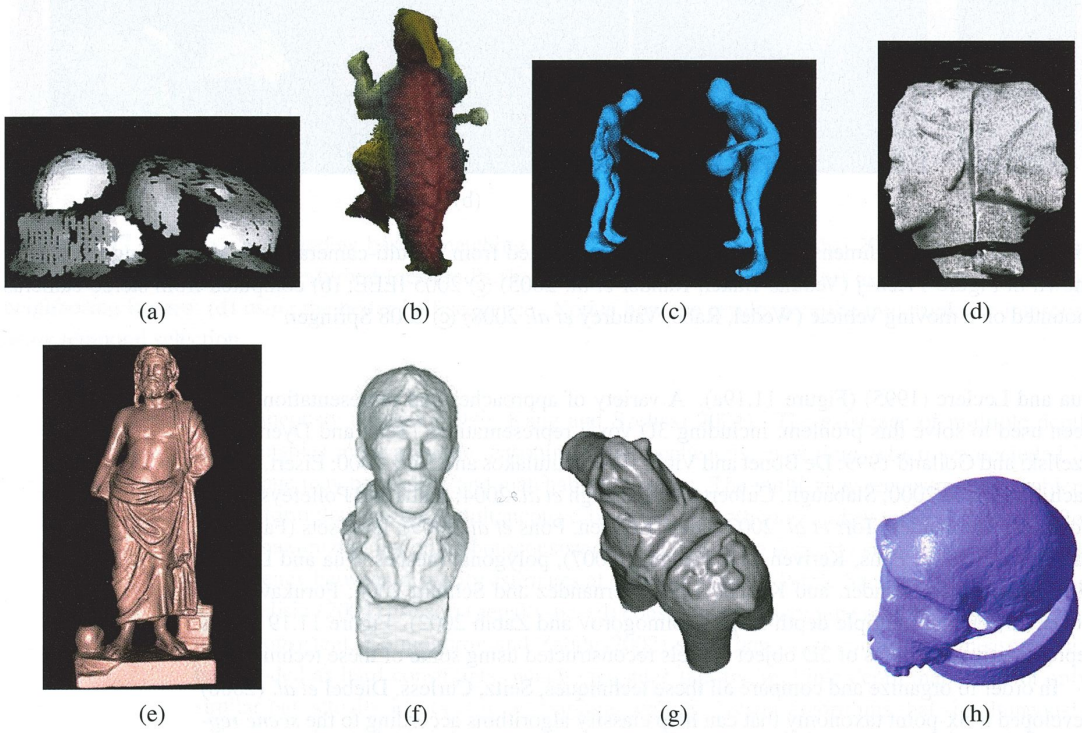


Figure 11.19 Multi-view stereo algorithms: (a) surface-based stereo (Fua and Leclerc 1995); (b) voxel coloring (Seitz and Dyer 1999) © 1999 Springer; (c) depth map merging (Narayanan, Rander, and Kanade 1998); (d) level set evolution (Faugeras and Keriven 1998) © 1998 IEEE; (e) silhouette and stereo fusion (Hernandez and Schmitt 2004) © 2004 Elsevier; (f) multi-view image matching (Pons, Keriven, and Faugeras 2005) © 2005 IEEE; (g) volumetric graph cut (Vogiatzis, Torr, and Cipolla 2005) © 2005 IEEE; (h) carved visual hulls (Furukawa and Ponce 2009) © 2009 Springer.

them into a 3D object model (Narayanan, Rander, and Kanade 1998; Furukawa and Ponce 2009; Goesele, Curless, and Seitz 2006; Goesele, Snavely, Curless *et al.* 2007; Furukawa, Curless, Seitz *et al.* 2010).

Photoconsistency measure. As we discussed in (Section 11.3.1), a variety of similarity measures can be used to compare pixel values in different images, including measures that try to discount illumination effects or be less sensitive to outliers. In multi-view stereo, algorithms have a choice of computing these measures directly on the surface of the model, i.e., in *scene space*, or projecting pixel values from one image (or from a textured model) back into another image, i.e., in *image space*. (The latter corresponds more closely to a Bayesian approach, since input images are noisy measurements of the colored 3D model.) The geometry of the object, i.e., its distance to each camera and its local surface normal, when available, can be used to adjust the matching windows used in the computation to account for foreshortening and scale change (Goesele, Snavely, Curless *et al.* 2007).

Visibility model. A big advantage that multi-view stereo algorithms have over single-depth-map approaches is their ability to reason in a principled manner about visibility and occlusions. Techniques that use the current state of the 3D model to predict which surface pixels are visible in each image (Kutulakos and Seitz 2000; Faugeras and Keriven 1998; Vogiatzis, Hernandez, Torr *et al.* 2007; Hiep, Keriven, Pons *et al.* 2009) are classified as using *geometric visibility models* in the taxonomy of Seitz, Curless, Diebel *et al.* (2006). Techniques that select a neighboring subset of image to match are called *quasi-geometric* (Narayanan, Rander, and Kanade 1998; Kang and Szeliski 2004; Hernandez and Schmitt 2004), while techniques that use traditional robust similarity measures are called *outlier-based*. While full geometric reasoning is the most principled and accurate approach, it can be very slow to evaluate and depends on the evolving quality of the current surface estimate to predict visibility, which can be a bit of a chicken-and-egg problem, unless conservative assumptions are used, as they are by Kutulakos and Seitz (2000).

Shape priors. Because stereo matching is often underconstrained, especially in textureless regions, most matching algorithms adopt (either explicitly or implicitly) some form of prior model for the expected shape. Many of the techniques that rely on optimization use a 3D smoothness or area-based photoconsistency constraint, which, because of the natural tendency of smooth surfaces to shrink inwards, often results in a *minimal surface* prior (Faugeras and Keriven 1998; Sinha and Pollefeys 2005; Vogiatzis, Hernandez, Torr *et al.* 2007). Approaches that carve away the volume of space often stop once a photoconsistent solution is found (Seitz and Dyer 1999; Kutulakos and Seitz 2000), which corresponds to a *maximal surface* bias, i.e., these techniques tend to over-estimate the true shape. Finally, multiple depth map approaches often adopt traditional *image-based* smoothness (regularization) constraints.

Reconstruction algorithm. The details of how the actual reconstruction algorithm proceeds is where the largest variety—and greatest innovation—in multi-view stereo algorithms can be found.

Some approaches use global optimization defined over a three-dimensional photoconsistency volume to recover a complete surface. Approaches based on graph cuts use polynomial complexity binary segmentation algorithms to recover the object model defined on the voxel grid (Sinha and Pollefeys 2005; Vogiatzis, Hernandez, Torr *et al.* 2007; Hiep, Keriven, Pons *et al.* 2009). Level set approaches use a continuous surface evolution to find a good minimum in the configuration space of potential surfaces and therefore require a reasonably good initialization (Faugeras and Keriven 1998; Pons, Keriven, and Faugeras 2007). In order for the photoconsistency volume to be meaningful, matching costs need to be computed in some robust fashion, e.g., using sets of limited views or by aggregating multiple depth maps.

An alternative approach to global optimization is to sweep through the 3D volume while computing both photoconsistency and visibility simultaneously. The *voxel coloring* algorithm of Seitz and Dyer (1999) performs a front-to-back plane sweep. On every plane, any voxels that are sufficiently photoconsistent are labeled as part of the object. The corresponding pixels in the source images can then be "erased", since they are already accounted for, and therefore do not contribute to further photoconsistency computations. (A similar approach, albeit without the front-to-back sweep order, is used by Szeliski and Golland (1999).) The resulting 3D volume, under noise- and resampling-free conditions, is guaranteed to produce

(a) (b) (c) (d) (e) (f)

Figure 11.20 The multi-view stereo data sets captured by Seitz, Curless, Diebel *et al.* (2006) © 2006 Springer. Only (a) and (b) are currently used for evaluation.

both a photoconsistent 3D model and to enclose whatever true 3D object model generated the images.

Unfortunately, voxel coloring is only guaranteed to work if all of the cameras lie on the same side of the sweep planes, which is not possible in general ring configurations of cameras. Kutulakos and Seitz (2000) generalize voxel coloring to *space carving*, where subsets of cameras that satisfy the voxel coloring constraint are iteratively selected and the 3D voxel grid is alternately carved away along different axes.

Another popular approach to multi-view stereo is to first independently compute multiple depth maps and then merge these partial maps into a complete 3D model. Approaches to depth map merging, which are discussed in more detail in Section 12.2.1, include signed distance functions (Curless and Levoy 1996), used by Goesele, Curless, and Seitz (2006), and Poisson surface reconstruction (Kazhdan, Bolitho, and Hoppe 2006), used by Goesele, Snavely, Curless *et al.* (2007). It is also possible to reconstruct sparser representations, such as 3D points and lines, and to interpolate them to full 3D surfaces (Section 12.3.1) (Taylor 2003).

Initialization requirements. One final element discussed by Seitz, Curless, Diebel *et al.* (2006) is the varying degrees of initialization required by different algorithms. Because some algorithms refine or evolve a rough 3D model, they require a reasonably accurate (or overcomplete) initial model, which can often be obtained by reconstructing a volume from object silhouettes, as discussed in Section 11.6.2. However, if the algorithm performs a global optimization (Kolev, Klodt, Brox *et al.* 2009; Kolev and Cremers 2009), this dependence on initialization is not an issue.

Empirical evaluation. In order to evaluate the large number of design alternatives in multi-view stereo, Seitz, Curless, Diebel *et al.* (2006) collected a dataset of calibrated images using a spherical gantry. A representative image from each of the six datasets is shown in Figure 11.20, although only the first two datasets have as yet been fully processed and used for evaluation. Figure 11.21 shows the results of running seven different algorithms on the *temple* dataset. As you can see, most of the techniques do an impressive job of capturing the fine details in the columns, although it is also clear that the techniques employ differing amounts of smoothing to achieve these results.

Since the publication of the survey by Seitz, Curless, Diebel *et al.* (2006), the field of

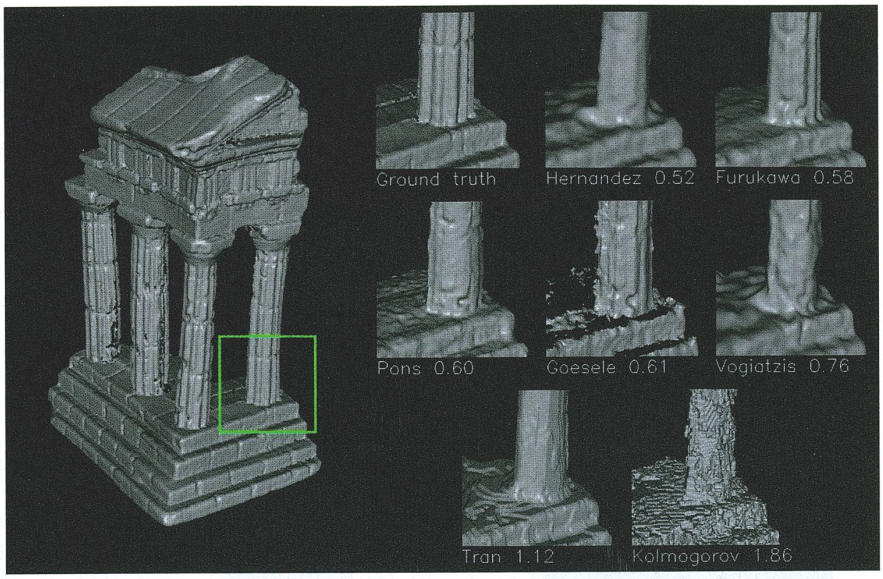


Figure 11.21 Reconstruction results (details) for seven algorithms (Hernandez and Schmitt 2004; Furukawa and Ponce 2009; Pons, Keriven, and Faugeras 2005; Goesele, Curless, and Seitz 2006; Vogiatzis, Torr, and Cipolla 2005; Tran and Davis 2002; Kolmogorov and Zabih 2002) evaluated by Seitz, Curless, Diebel *et al.* (2006) on the 47-image Temple Ring dataset. The numbers underneath each detail image are the accuracy of each of these techniques measured in millimeters.

multi-view stereo has continued to advance at a rapid pace (Strecha, Fransens, and Van Gool 2006; Hernandez, Vogiatzis, and Cipolla 2007; Habbecke and Kobbelt 2007; Furukawa and Ponce 2007; Vogiatzis, Hernandez, Torr *et al.* 2007; Goesele, Snavely, Curless *et al.* 2007; Sinha, Mordohai, and Pollefeys 2007; Gargallo, Prados, and Sturm 2007; Merrell, Akbarzadeh, Wang *et al.* 2007; Zach, Pock, and Bischof 2007b; Furukawa and Ponce 2008; Hornung, Zeng, and Kobbelt 2008; Bradley, Boubekeur, and Heidrich 2008; Zach 2008; Campbell, Vogiatzis, Hernández *et al.* 2008; Kolev, Klodt, Brox *et al.* 2009; Hiep, Keriven, Pons *et al.* 2009; Furukawa, Curless, Seitz *et al.* 2010). The multi-view stereo evaluation site, http://vision.middlebury.edu/mview/, provides quantitative results for these algorithms along with pointers to where to find these papers.

11.6.2 Shape from silhouettes

In many situations, performing a foreground–background segmentation of the object of interest is a good way to initialize or fit a 3D model (Grauman, Shakhnarovich, and Darrell 2003; Vlasic, Baran, Matusik *et al.* 2008) or to impose a convex set of constraints on multi-view stereo (Kolev and Cremers 2008). Over the years, a number of techniques have been developed to reconstruct a 3D volumetric model from the intersection of the binary silhouettes projected into 3D. The resulting model is called a *visual hull* (or sometimes a *line hull*), analogous with the convex hull of a set of points, since the volume is maximal with respect

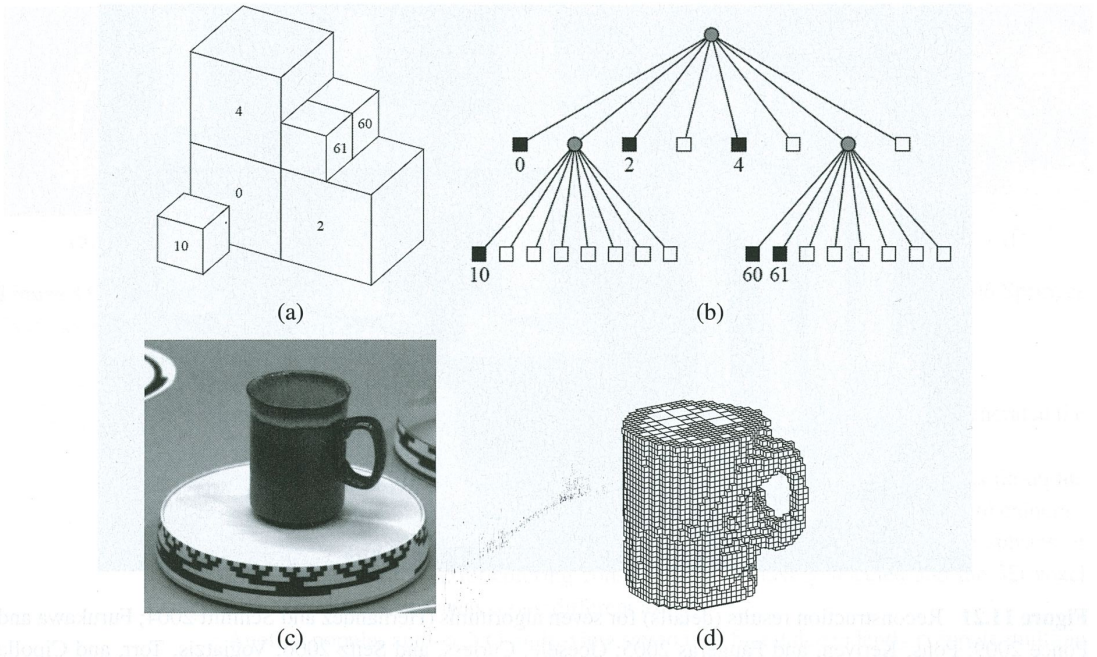


Figure 11.22 Volumetric octree reconstruction from binary silhouettes (Szeliski 1993) © 1993 Elsevier: (a) octree representation and its corresponding (b) tree structure; (c) input image of an object on a turntable; (d) computed 3D volumetric octree model.

to the visual silhouettes and surface elements are tangent to the viewing rays (lines) along the silhouette boundaries (Laurentini 1994). It is also possible to carve away a more accurate reconstruction using multi-view stereo (Sinha and Pollefeys 2005) or by analyzing cast shadows (Savarese, Andreetto, Rushmeier *et al.* 2007).

Some techniques first approximate each silhouette with a polygonal representation and then intersect the resulting faceted conical regions in three-space to produce polyhedral models (Baumgart 1974; Martin and Aggarwal 1983; Matusik, Buehler, and McMillan 2001), which can later be refined using triangular splines (Sullivan and Ponce 1998). Other approaches use voxel-based representations, usually encoded as octrees (Samet 1989), because of the resulting space–time efficiency. Figures 11.22a–b show an example of a 3D octree model and its associated colored tree, where black nodes are interior to the model, white nodes are exterior, and gray nodes are of mixed occupancy. Examples of octree-based reconstruction approaches include those by Potmesil (1987), Noborio, Fukada, and Arimoto (1988), Srivasan, Liang, and Hackwood (1990), and Szeliski (1993).

The approach of Szeliski (1993) first converts each binary silhouette into a one-sided variant of a distance map, where each pixel in the map indicates the largest square that is completely inside (or outside) the silhouette. This makes it fast to project an octree cell into the silhouette to confirm whether it is completely inside or outside the object, so that it can be colored black, white, or left as gray (mixed) for further refinement on a smaller grid. The octree construction algorithm proceeds in a coarse-to-fine manner, first building an

octree at a relatively coarse resolution, and then refining it by revisiting and subdividing all the input images for the gray (mixed) cells whose occupancy has not yet been determined. Figure 11.22d shows the resulting octree model computed from a coffee cup rotating on a turntable.

More recent work on visual hull computation borrows ideas from image-based rendering, and is hence called an *image-based visual hull* (Matusik, Buehler, Raskar *et al.* 2000). Instead of precomputing a global 3D model, an image-based visual hull is recomputed for each new viewpoint, by successively intersecting viewing ray segments with the binary silhouettes in each image. This not only leads to a fast computation algorithm but also enables fast texturing of the recovered model with color values from the input images. This approach can also be combined with high-quality deformable templates to capture and re-animate whole body motion (Vlasic, Baran, Matusik *et al.* 2008).

11.7 Additional reading

The field of stereo correspondence and depth estimation is one of the oldest and most widely studied topics in computer vision. A number of good surveys have been written over the years (Marr and Poggio 1976; Barnard and Fischler 1982; Dhond and Aggarwal 1989; Scharstein and Szeliski 2002; Brown, Burschka, and Hager 2003; Seitz, Curless, Diebel *et al.* 2006) and they can serve as good guides to this extensive literature.

Because of computational limitations and the desire to find appearance-invariant correspondences, early algorithms often focused on finding *sparse correspondences* (Hannah 1974; Marr and Poggio 1979; Mayhew and Frisby 1980; Baker and Binford 1981; Arnold 1983; Grimson 1985; Ohta and Kanade 1985; Bolles, Baker, and Marimont 1987; Matthies, Kanade, and Szeliski 1989; Hsieh, McKeown, and Perlant 1992; Bolles, Baker, and Hannah 1993).

The topic of computing epipolar geometry and pre-rectifying images is covered in Sections 7.2 and 11.1 and is also treated in textbooks on multi-view geometry (Faugeras and Luong 2001; Hartley and Zisserman 2004) and articles specifically on this topic (Torr and Murray 1997; Zhang 1998a,b). The concepts of the *disparity space* and *disparity space image* are often associated with the seminal work by Marr (1982) and the papers of Yang, Yuille, and Lu (1993) and Intille and Bobick (1994). The plane sweep algorithm was first popularized by Collins (1996) and then generalized to a full arbitrary projective setting by Szeliski and Golland (1999) and Saito and Kanade (1999). Plane sweeps can also be formulated using cylindrical surfaces (Ishiguro, Yamamoto, and Tsuji 1992; Kang and Szeliski 1997; Shum and Szeliski 1999; Li, Shum, Tang *et al.* 2004; Zheng, Kang, Cohen *et al.* 2007) or even more general topologies (Seitz 2001).

Once the topology for the cost volume or DSI has been set up, we need to compute the actual photoconsistency measures for each pixel and potential depth. A wide range of such measures have been proposed, as discussed in Section 11.3.1. Some of these are compared in recent surveys and evaluations of matching costs (Scharstein and Szeliski 2002; Hirschmüller and Scharstein 2009).

To compute an actual depth map from these costs, some form of optimization or selection criterion must be used. The simplest of these are sliding windows of various kinds, which are discussed in Section 11.4 and surveyed by Gong, Yang, Wang *et al.* (2007) and Tombari,

Mattoccia, Di Stefano *et al.* (2008). More commonly, global optimization frameworks are used to compute the best disparity field, as described in Section 11.5. These techniques include dynamic programming and truly global optimization algorithms, such as graph cuts and loopy belief propagation. Because the literature on this is so extensive, it is described in more detail in Section 11.5. A good place to find pointers to the latest results in this field is the Middlebury Stereo Vision Page at http://vision.middlebury.edu/stereo.

Algorithms for multi-view stereo typically fall into two categories. The first include algorithms that compute traditional depth maps using several images for computing photoconsistency measures (Okutomi and Kanade 1993; Kang, Webb, Zitnick *et al.* 1995; Nakamura, Matsuura, Satoh *et al.* 1996; Szeliski and Golland 1999; Kang, Szeliski, and Chai 2001; Vaish, Szeliski, Zitnick *et al.* 2006; Gallup, Frahm, Mordohai *et al.* 2008). Optionally, some of these techniques compute multiple depth maps and use additional constraints to encourage the different depth maps to be consistent (Szeliski 1999; Kolmogorov and Zabih 2002; Kang and Szeliski 2004; Maitre, Shinagawa, and Do 2008; Zhang, Jia, Wong *et al.* 2008).

The second category consists of papers that compute true 3D volumetric or surface-based object models. Again, because of the large number of papers published on this topic, rather than citing them here, we refer you to the material in Section 11.6.1, the survey by Seitz, Curless, Diebel *et al.* (2006), and the on-line evaluation Web site at http://vision.middlebury.edu/mview/.

11.8 Exercises

Ex 11.1: Stereo pair rectification Implement the following simple algorithm (Section 11.1.1):

1. Rotate both cameras so that they are looking perpendicular to the line joining the two camera centers $\boldsymbol{c}_0$ and $\boldsymbol{c}_1$. The smallest rotation can be computed from the cross product between the original and desired optical axes.
2. Twist the optical axes so that the horizontal axis of each camera looks in the direction of the other camera. (Again, the cross product between the current x-axis after the first rotation and the line joining the cameras gives the rotation.)
3. If needed, scale up the smaller (less detailed) image so that it has the same resolution (and hence line-to-line correspondence) as the other image.

Now compare your results to the algorithm proposed by Loop and Zhang (1999). Can you think of situations where their approach may be preferable?

Ex 11.2: Rigid direct alignment Modify your spline-based or optical flow motion estimator from Exercise 8.4 to use epipolar geometry, i.e. to only estimate disparity.

(Optional) Extend your algorithm to simultaneously estimate the epipolar geometry (without first using point correspondences) by estimating a base homography corresponding to a reference plane for the dominant motion and then an epipole for the residual parallax (motion).

Ex 11.3: Shape from profiles Reconstruct a surface model from a series of edge images (Section 11.2.1).

1. Extract edges and link them (Exercises 4.7–4.8).
2. Based on previously computed epipolar geometry, match up edges in triplets (or longer sets) of images.
3. Reconstruct the 3D locations of the curves using osculating circles (11.5).
4. Render the resulting 3D surface model as a sparse mesh, i.e., drawing the reconstructed 3D profile curves and links between 3D points in neighboring images with similar osculating circles.

Ex 11.4: Plane sweep Implement a plane sweep algorithm (Section 11.1.2).

If the images are already pre-rectified, this consists simply of shifting images relative to each other and comparing pixels. If the images are not pre-rectified, compute the homography that resamples the target image into the reference image's coordinate system for each plane.

Evaluate a subset of the following similarity measures (Section 11.3.1) and compare their performance by visualizing the disparity space image (DSI), which should be dark for pixels at correct depths:

- squared difference (SD);
- absolute difference (AD);
- truncated or robust measures;
- gradient differences;
- rank or census transform (the latter usually performs better);
- mutual information from a pre-computed joint density function.

Consider using the Birchfield and Tomasi (1998) technique of comparing ranges between neighboring pixels (different shifted or warped images). Also, try pre-compensating images for bias or gain variations using one or more of the techniques discussed in Section 11.3.1.

Ex 11.5: Aggregation and window-based stereo Implement one or more of the matching cost aggregation strategies described in Section 11.4:

- convolution with a box or Gaussian kernel;
- shifting window locations by applying a min filter (Scharstein and Szeliski 2002);
- picking a window that maximizes some match-reliability metric (Veksler 2001, 2003);
- weighting pixels by their similarity to the central pixel (Yoon and Kweon 2006).

Once you have aggregated the costs in the DSI, pick the winner at each pixel (winner-take-all), and then optionally perform one or more of the following post-processing steps:

1. compute matches both ways and pick only the reliable matches (draw the others in another color);
2. tag matches that are unsure (whose confidence is too low);

3. fill in the matches that are unsure from neighboring values;
4. refine your matches to sub-pixel disparity by either fitting a parabola to the DSI values around the winner or by using an iteration of Lukas–Kanade.

Ex 11.6: Optimization-based stereo Compute the disparity space image (DSI) volume using one of the techniques you implemented in Exercise 11.4 and then implement one (or more) of the global optimization techniques described in Section 11.5 to compute the depth map. Potential choices include:

- dynamic programming or scanline optimization (relatively easy);
- semi-global optimization (Hirschmüller 2008), which is a simple extension of scanline optimization and performs well;
- graph cuts using alpha expansions (Boykov, Veksler, and Zabih 2001), for which you will need to find a max-flow or min-cut algorithm (http://vision.middlebury.edu/stereo);
- loopy belief propagation (Appendix B.5.3).

Evaluate your algorithm by running it on the Middlebury stereo data sets.

How well does your algorithm do against local aggregation (Yoon and Kweon 2006)? Can you think of some extensions or modifications to make it even better?

Ex 11.7: View interpolation, revisited Compute a dense depth map using one of the techniques you developed above and use it (or, better yet, a depth map for each source image) to generate smooth in-between views from a stereo data set.

Compare your results against using the ground truth depth data (if available).

What kinds of artifacts do you see? Can you think of ways to reduce them?

More details on implementing such algorithms can be found in Section 13.1 and Exercises 13.1–13.4.

Ex 11.8: Multi-frame stereo Extend one of your previous techniques to use multiple input frames (Section 11.6) and try to improve the results you obtained with just two views.

If helpful, try using temporal selection (Kang and Szeliski 2004) to deal with the increased number of occlusions in multi-frame data sets.

You can also try to simultaneously estimate multiple depth maps and make them consistent (Kolmogorov and Zabih 2002; Kang and Szeliski 2004).

Test your algorithms out on some standard multi-view data sets.

Ex 11.9: Volumetric stereo Implement voxel coloring (Seitz and Dyer 1999) as a simple extension to the plane sweep algorithm you implemented in Exercise 11.4.

1. Instead of computing the complete DSI all at once, evaluate each plane one at a time from front to back.
2. Tag every voxel whose photoconsistency is below a certain threshold as being part of the object and remember its average (or robust) color (Seitz and Dyer 1999; Eisert, Steinbach, and Girod 2000; Kutulakos 2000; Slabaugh, Culbertson, Slabaugh *et al.* 2004).

3. Erase the input pixels corresponding to tagged voxels in the input images, e.g., by setting their alpha value to 0 (or to some reduced number, depending on occupancy).

4. As you evaluate the next plane, use the source image alpha values to modify your photoconsistency score, e.g., only consider pixels that have full alpha or weight pixels by their alpha values.

5. If the cameras are not all on the same side of your plane sweeps, use space carving (Kutulakos and Seitz 2000) to cycle through different subsets of source images while carving away the volume from different directions.

Ex 11.10: Depth map merging Use the technique you developed for multi-frame stereo in Exercise 11.8 or a different technique, such as the one described by Goesele, Snavely, Curless *et al.* (2007), to compute a depth map for every input image.

Merge these depth maps into a coherent 3D model, e.g., using Poisson surface reconstruction (Kazhdan, Bolitho, and Hoppe 2006).

Ex 11.11: Shape from silhouettes Build a silhouette-based volume reconstruction algorithm (Section 11.6.2). Use an octree or some other representation of your choosing.

Humans perceive the three-dimensional structure of the world with apparent ease. However, despite all of the recent advances in computer vision research, the dream of having a computer interpret an image at the same level as a two-year old remains elusive. Why is computer vision such a challenging problem and what is the current state of the art?

Computer Vision: Algorithms and Applications explores the variety of techniques commonly used to analyze and interpret images. It also describes challenging real-world applications where vision *is* being successfully used, both for specialized applications such as medical imaging, and for fun, consumer-level tasks such as image editing and stitching, which students can apply to their own personal photos and videos.

More than just a source of "recipes," this exceptionally authoritative and comprehensive textbook/reference also takes a scientific approach to basic vision problems, formulating physical models of the imaging process before inverting them to produce descriptions of a scene. These problems are also analyzed using statistical models and solved using rigorous engineering techniques.

Topics and Features:

- Structured to support active curricula and project-oriented courses, with tips in the Introduction for using the book in a variety of customized courses
- Presents exercises at the end of each chapter with a heavy emphasis on testing algorithms and containing numerous suggestions for small mid-term projects
- Provides additional material and more detailed mathematical topics in the Appendices, which cover linear algebra, numerical techniques, and Bayesian estimation theory
- Suggests additional reading at the end of each chapter, including the latest research in each sub-field, in addition to a full Bibliography at the end of the book
- Supplies supplementary course material for students at the associated website, http://szeliski.org/Book/

Suitable for an upper-level undergraduate or graduate-level course in computer science or engineering, this textbook focuses on basic techniques that work under real-world conditions and encourages students to push their creative boundaries. Its design and exposition also make it eminently suitable as a unique reference to the fundamental techniques and current research literature in computer vision.

Dr. Richard Szeliski has more than 25 years' experience in computer vision research, most notably at Digital Equipment Corporation and Microsoft Research. This text draws on that experience, as well as on computer vision courses he has taught at the University of Washington and Stanford.

ISBN 978-1-84882-934-3

springer.com