l (H.T.S.) in Enschede,
: B.Sc. degree in elec-
and the Ir. degree in
1975 from the Univer
The Netherlands.
n working at the Uni-
al levels of responsibil-
senior staff member
nt") in the Laboratory
University of Twente,
r research and educa-

A Chip Set for Lossless Image Compression

Imran Ali Shah, Olu Akiwumi-Assani, and Brian Johnson

Abstract —**This chip set performs the *S*-transform [1], [2] image decomposition and the Lempel–Ziv [L–Z] [5], [6], [9] type of entropy coding. The compression ratio achieved by the system exceeds 2:1 for 10-b 2000×2000-pixel computed radiographs [3] at an average rate of 7.5 Mpixels/s [4]. The paper presents the transform and coding algorithms, the main architectural features of the chips, and outlines some performance specifications.**

I. Introduction

As APPLICATIONS of digital images proliferate, the cost of storing and transmitting the image data becomes a common problem. Lossless image compression techniques provide a general-purpose solution free of compromise. The compressed image is stored or transmitted in typically less than half the space or less than half the time of the original. Lossless compression can be achieved by a transform process followed by entropy coding. We have designed two application-specific integrated circuits (ASIC's) to perform the *S* transform [1], [2]: a hierarchical transformation, and a data compressor/decompressor for Lempel–Ziv entropy coding [5], [6]. The chips may be used independently or together for image compression. One of the applications of such a chip set is in the medical PACS environment [7]. The algorithms and chip architectures will be described. Some of the important performance parameters will also be discussed.

II. *S*-Transform Processor

The principal objective of the transform process is to achieve a spectral distribution of a given source signal such that most of its energy is concentrated in a subset of the coefficients in the transform domain. If the signal source exhibits a high degree of correlation between successive samples, then it follows that for a given transform the signal energy is concentrated in a narrow spectral range. In practice, the transforms normally applied to image coding trade off packing (decorrelation) efficiency with implementation complexity. The *S* transform is computationally simple, and provides a reasonable reduction of redundancy in the image. The decorrelation induced by the *S* transform was theoretically analyzed by Lux [1]. In [3], this was experimentally done by computing the zeroth-order entropy

$$E = -\sum_{i=1}^{N} p_i \log_2(p_i)$$

before and after applying the transform to a set of 15 computed radiograph (CR) images. N refers to the number of digitization levels, and p_i the probability of occurrence. It was observed that the average entropy of the images was reduced from 8.8 to 4.6 b. The *S* transform is not perfect or equal to the Karhunen–Loeve transform, but it offers a good trade-off between performance and simplicity of implementation. The transform also provides a means for hierarchial image representation, and has been used for some image enhancements [8].

A. S-Transform Algorithm

The *S* transform is the Hadamard transform for 2×2-pixel image blocks. To apply the *S* transform, an image must have a size of $2n \times 2m$ pixels or be appropriately padded.

If a, b, c, and d are the pixels in the 2×2 block as shown:

$$\begin{matrix} .a & .b \\ .c & .d \end{matrix}$$

then the forward transform coefficients are given by (1):

$$\begin{aligned} \Sigma &= a+b+c+d \\ \Delta_H &= a-b+c-d \\ \Delta_V &= a+b-c-d \\ \Delta_D &= a-b-c+d \end{aligned} \tag{1}$$

where Σ is the sum coefficient and the Δ coefficients represent horizontal, vertical, and diagonal spatial frequency components of the image.

The *S* transform provides a means to hierarchically decompose a large image. To achieve this, the 2×2 blocks of the image are transformed according to (1), resulting in four subimages, one for each coefficient. Each of the subimages has half the spatial resolution of its original. This transformation shifts most of the energy of the 2×2 pixel block into the Σ coefficient; thus the distribution for the Δ images is narrow and requires fewer bits for coding, permitting better compression. To achieve hierarchical decomposition of an image, the lower spatial resolution Σ image at each step is further transformed while the three Δ images are entropy-coded for storage or transmission. This decomposition step can be successively applied until a basis image of much lower resolution is achieved. This results in a representation consisting of a basis image, followed by layers of coded Δ images of increasing spatial resolution (see Fig. 1). To reconstruct higher resolution images, the Σ image of a given layer is recombined with its corresponding Δ's according to the inverse transform equations.

In our implementation of the *S* transform, the forward transform is computed by a slightly modified set of equa-

Manuscript received August 17, 1990; revised November 12, 1990.
I. A. Shah was with Philips Laboratories, North American Philips Corporation, Briarcliff Manor, NY 10510. He is now with Philips Research Laboratories, 5600 JA Eindhoven, The Netherlands.
O. Akiwumi-Assani and B. Johnson are with Philips Laboratories, North American Philips Corporation, Briarcliff Manor, NY 10510.
IEEE Log Number 9041909.

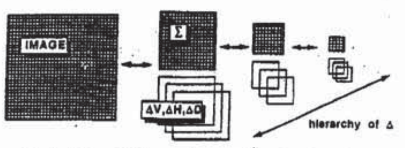


Fig. 1. Hierarchical image decomposition using S transform.

tions:

$$\begin{aligned}\Sigma &= [a+b+c+d]/4\\ \Delta_H &= [(a+c)-(b+d)]/2\\ \Delta_V &= [(a+b)-(c+d)]/2\\ \Delta_D &= a-b-c+d. \qquad (2)\end{aligned}$$

Given the transform coefficients, the original pixels can be obtained by the inverse transform realized by

$$\begin{aligned}a &= [(2\cdot s1+F[s2])+s2]/2\\ b &= [(2\cdot s1+F[s2])-s2]/2\\ c &= [(2\cdot d1+F[d2])+d2]/2\\ d &= [(2\cdot d1+F[d2])-d2]/2 \qquad (3)\end{aligned}$$

where the intermediate variables $s1$, $d1$, $s2$, and $d2$ are

$$\begin{aligned}s1 &= [(2\cdot\Sigma+F[\Delta_V])+\Delta_V]/2\\ d1 &= [(2\cdot\Sigma+F[\Delta_V])-\Delta_V]/2\\ s2 &= [(2\cdot\Delta_H+F[\Delta_D])+\Delta_D]/2\\ d2 &= [(2\cdot\Delta_H+F[\Delta_D])-\Delta_D]/2 \qquad (4)\end{aligned}$$

and the function $F[\cdot]$ returns the least significant bit of its argument. This form of the forward transform equation was chosen to ensure that the dynamic range of the pixels in any Σ image remains the same regardless of the number of hierarchical decompositions (or reconstructions) an image is subject to. There is no loss of significant bits because the least-significant-bit information is carried in the other coefficients. By means of the $F[\cdot]$ function in the inverse transform, all variables are resolved to their full resolution prior to any computation. Given that the pixels in the original image are digitized to b bits (positive integers), the Σ coefficient will remain b bits (positive integer), the Δ_H and Δ_V coefficients will each be $b+1$ bits (signed integers), and the Δ_D coefficient $b+2$ bits (signed integer).

B. S-Transform Chip Architecture

Fig. 2 shows the architecture of the S-transform pipeline processor (SPIPE) chip. The device has three ports for pixels, coefficients, and the microprocessor interface. Data transfer through the pixel and coefficient ports is controlled by a self-regulating synchronous protocol (PCB). The processor consists of one array of four arithmetic units that implement addition, subtraction, multiplication, and division by 2. Two feedback data buses ($FB1$ and $FB2$) make this architecture optimum for the computation of both the forward as well as the inverse transform. The short feedback bus, $FB1$, enables 100% utilization of the processor. In inverse mode, the block Mux & Limiter ensures correctly reconstructed pixel intensity values. The block Tribuf provides isolation between internal and external buses. The PCB blocks implement the data transfer protocol between the pipeline registers.

The pixel port consists of two 13-b pixel buses for the raster lines $[a,b]$ and $[c,d]$ with accompanying pipeline control signals. Thus, this port behaves as if connected to two merging pipeline sources (forward transform) or to two branching pipeline outputs (inverse transform). Logically, the coefficient port behaves in a similar fashion. However, because the destinations of the Σ and the Δ_V coefficients are not always the same (Δ's go to the coder), we really have three pipeline sources (sinks) and associated controls. The $[\Sigma, \Delta_V]$ bus is 14 b wide to accommodate the larger values of the Δ_v coefficient. Similarly, the $[\Delta_H, \Delta_D]$ bus is 15 b wide. Time multiplexing of the pipeline input/output buses and the inherent two-phase computation of the S transform enable the device to maintain a steady rate of operation at the system clock frequency in either forward or inverse mode. The PCB's on the inputs and outputs of the device ensure that the correct synchronization of the data from or to the two logical pipeline branches is guaranteed. The microprocessor port provides a 4-b bidirectional data bus and control signals.

In forward mode, the pixel port is the input to the device while the coefficient port is the output, and the lower half of the data path is inactive. The effective internal pipeline latency is four clock cycles. Only the short feedback path is active and is used every other cycle to implement a two-stage butterfly computation. It takes two clock cycles to acquire all four pixels in the matrix, and also two cycles to output the four coefficients. Thus the total input-to-output latency is six cycles. In inverse mode, all elements of the data path are operative, introducing an additional cycle to the pipeline latency. In this case, the input to the device is the coefficient port, while the output is the pixel port. Again two clock cycles are required to fully load all four coefficients, which are then transferred over the longer feedback path, $FB2$, to the input-stage Switch & Register. The output from the processor is passed through the lower half of the data path. Again two cycles are required to output the reconstructed pixels.

Two 4-b programmable registers define the operating modes and parameters. Register 1 defines the principal modes of operation of the device: forward transform, inverse transform, forward bypass, and inverse bypass. The bypass modes are provided to enable direct transfer of pixels through the chip.

One of two scan chains can be selected to support diagnostics. The control signals clock, reset, and test enable the device to be placed into the principal, diagnostic, or reset state. Upon power-up or a reset condition, the device is automatically placed into the forward transform mode of operation.

By using one processor, feedback data buses, and time multiplexing of the pipeline I/O, we have been able to achieve an efficient VLSI implementation, yet one that naturally fits the raster format of scanned images. The chip is based on a standard cell design, in a 1.5-μm CMOS double-metal process. It will be available packaged as a 84-pin PLCC component operating at 5 V requiring no more than 600 mW at 10 MHz. At this clock frequency, the chip can sustain a processing rate of 20 million pixels per second.

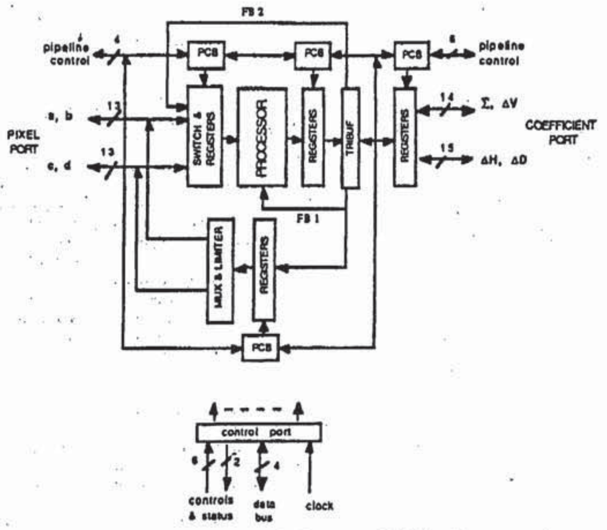


Fig. 2. Simplified *S* transform pipeline processor (SPIPE) architecture.

III. Data Compressor Decompressor

The data compressor/decompressor (DCD) uses a version of the Lempel–Ziv (L–Z) compression algorithm [5], [9]. The L–Z algorithm replaces input strings with code words, creating a compressed output. A code table, containing the data string that each code represents, is constructed based on the data encountered during compression. During decompression, the same code table is recreated from the sequence of code words. In the table, strings are represented by the code for a shorter string plus an additional data word. During encoding, a string of a single element will always be found. The next longer string represented by that string plus the next data word will be searched for. This process continues recursively until the longer string is not found in the table. In this case, the longer string is entered into the table and the code for the shorter string is transmitted.

While encoding, a transmission is made after every failure and is accompanied by the creation of a new code-table entry. This knowledge is used by the decoder. Each time the decoder receives a code, it makes a corresponding entry to its table by appending the first character of the currently received code to the previous code and assigning a new code word to it in the code table. At the same time, it outputs the string corresponding to the current code, always operating just a step behind the encoder.

Ziv and Lempel [5] have derived upper bounds for the compression ratio attainable with full *a priori* knowledge of the data by fixed code-table schemes. They have then proceeded to show that the efficiency of their code with no *a priori* knowledge of the data approaches those bounds.

A. Solving the L–Z Search Requirement

When in encoding, a new data word is received, a prefix is appended to it, and the code table is searched for a code word representing this combination. In the DCD, we limited the code word size to 16 b, and hence have to search through 64K locations. Hashing [10]–[13] was employed to perform the high-speed search and retrieval.

B. Hashing

Hashing is a technique designed to speed the retrieval of data associated with a key K [10]. The range of K is usually extremely large; hence K cannot be directly used as an index or address to find the data. One computes a function $f(K)$, the hash function, which is the location of K and the associated data in the table. The hash function provides a mapping from the large range of K to a smaller range. Such a mapping is not unique; more than one entry can be mapped onto the same location. This is called a collision. The difference between hashing schemes lies in the way they resolve collisions.

The schemes trade off memory for performance. The two main categories of collision resolution are "chaining" and "open addressing." In chaining, a link is used at each table entry to point to a chained list of additional entries in an overflow memory that had been mapped earlier to the same location. In open addressing, the additional entries are placed within the same table by some predefined method. In chaining, the mapping is preserved, while in open addressing, the mapping is lost.

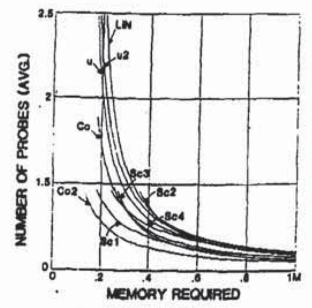


Fig. 3. Average number of memory probes versus memory required (in megabits).

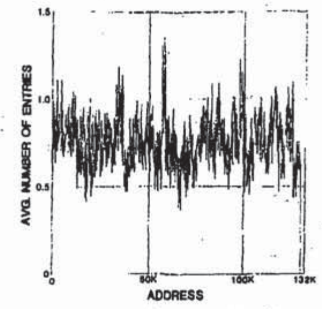


Fig. 4. Average number of entries at each address (chosen hash matrix).

In the "open addressing" class of hashing schemes, uniform hashing has been shown to be optimal [12], while coalesced hashing is shown optimal [11] in the "chaining" class of hashing schemes. We analyzed the performance of nine different collision resolution schemes: linear (LIN, [10]), uniform (u, u2, [10], [12]), coalesced (Co, Co2, [10], [11]), and separate chaining (Sc1, Sc2, Sc3, Sc4, [10], [11], [13]). We did not consider the commonly used performance measure—number of memory probes versus load-factor (fraction of the table filled)—as an important measure, since the *same* load factor in different schemes represents *different* amounts of memory required and, hence, is not a good measure of relative performance. The performance measure we used was the average number of memory probes for resolving a success and a failure as a function of *total memory employed*. The outcome of the analysis is shown in Fig. 3. From the performance and implementation point of view, separate chaining 1 (Sc1) was most attractive. In chaining, the address of the overflow memory implicitly contains the code word associated with the key and we do not need to store it, thus saving memory.

C. Memory Saving: Abbreviated Keys and Pipelining

Since the mapping was preserved by chaining, the address of the key gives information about the key and only a smaller part of it needs to be stored. For example, if the hash function is a division, then the quotient can be used as the address and the remainder stored as the abbreviated key.

D. Hash Function

The performance of any hashing scheme depends on the hash function. A good hash function spreads all the input keys uniformly across the memory range. Additional constraints were placed due to the desired use of abbreviated keys and simple hardware. A hash function based on binary matrix multiplication was developed, which satisfies all the above constraints. The hash function and abbreviated key generator each consist of an array of EXCLUSIVE-OR gates which combine bits of the input key to form the hash function and abbreviated key.

It is convenient to represent the generators as matrices (H, A) of ONE's and ZERO's, and the input key, hash function, and abbreviated key as vectors $(\vec{k}, \vec{h}, \vec{a})$ of ONE's and ZERO's. With this representation, the generators for the hash function and abbreviated key may be expressed as

$$\vec{h} = H\vec{k} \tag{5}$$

$$\vec{a} = A\vec{k} \tag{6}$$

where binary multiplication is performed by ANDing of bits, and binary addition is performed by EXCLUSIVE ORing of bits. If the hash function generator matrix H is randomly created, the resulting hash function has good properties. The difficulty is in creating an associated abbreviated key generator.

We combine (5) and (6) into one equation and obtain

$$\begin{vmatrix} h \\ a \end{vmatrix} = \begin{vmatrix} H \\ A \end{vmatrix} \vec{k}. \tag{7}$$

As long as the combined matrix

$$\begin{vmatrix} H \\ A \end{vmatrix}$$

is invertible, the combination of hash function h and abbreviated key a uniquely represents the original key k as required, since k may be computed from h and a as shown below:

$$k = \begin{vmatrix} H \\ A \end{vmatrix}^{-1} \begin{vmatrix} h \\ a \end{vmatrix}. \tag{8}$$

This allows us to create a combined pseudorandom matrix, restricted to invertible matrices, which will result in a generator having good statistical properties for the hash function and valid abbreviated keys.

Once the combined matrix is determined, it may be separated into the individual matrices H and A to generate the hash function and abbreviated key, respectively.

Extensive software simulations using representative data were performed to select the binary matrix. Fig. 4 shows the average number of entries at each memory location for the chosen matrix. It shows the uniform spreading of the input keys across the memory range. Simulations confirm our theoretical results of less than 1.2 probes per key. As a contrast, Fig. 5 shows a different matrix used as the hash function. It is easy to see that memory utilization is not as uniform, resulting in many "clusters" of over- and under-

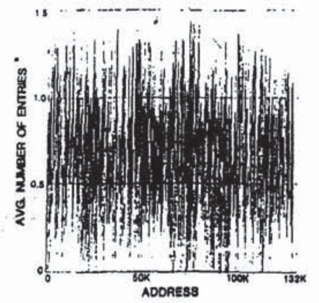


Fig. 5. Average number of entries at each address.

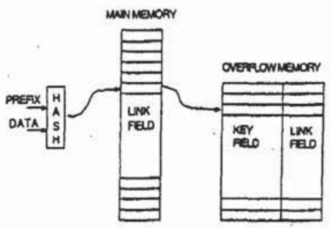


Fig. 6. Reduced memory architecture.

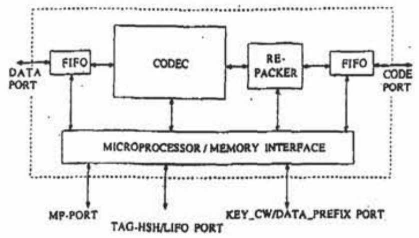


Fig. 7. Simplified DCD block diagram.

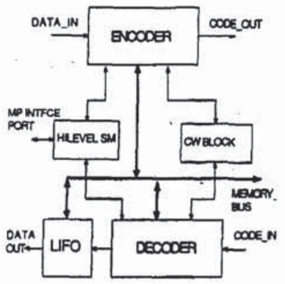


Fig. 8. Codec block of the DCD.

utilized locations. This hash function requires more than four probes in resolving each key.

Since all the algebra involved in the above binary operations is modulo two, the hardware is simple, consisting only of EXCLUSIVE-OR gates.

Additional memory savings are obtained when the main memory contains only a link field. This field points to the overflow memory containing the key, the implicit code word, and further link fields. Refer to Fig. 6. K is hashed to the link table (main memory) by the hash function. If there is no link at that location, then we have a failure and a link is created to the next available empty location in the overflow memory where K is inserted. On the other hand, if there is a link at that location, we follow it and look for K in the chained list of the overflow memory.

A disadvantage of the reduced main memory scheme is that a minimum of two memory accesses is required to find each K. This problem can be overcome by pipelining the two accesses.

E. Algorithm Implementation with Hashing

When the idea of pipelining main and overflow memory accesses is extended to the L–Z algorithm, a potential problem arises. The main memory cannot be accessed without the prefix and the prefix will not be available without accessing and resolving the search in the overflow memory. However, inherent properties of the algorithm do allow us to pipeline it.

Simulations [3] have shown that the L–Z algorithm compresses an image by a factor of 2 to 3, with average string lengths of between 3 and 4. Therefore, while searching for a prefix–data combination, success will occur three times as often as failure. With this knowledge, we can anticipate a prefix *before* the search is resolved and we can be correct about 75% of the time. If we anticipated incorrectly, we disregard the incorrect link and use the next anticipated prefix. This backtracking method allows efficient pipelining of the main (link) and overflow (key and code word) memory accesses to improve speed.

F. Chip Architecture and Features

The DCD chip consists of two 16-b FIFO's, a codec (including encoder, decoder, LIFO (necessary for decoding), code-word generation block, and a high-level controller), a repacker, and a microprocessor interface section (Fig. 7). The bidirectional FIFO's are to smooth the bursty data rates inherent in the compression and decompression processes. During coding, the decoder is idle, and during decoding, the encoder is idle. The bidirectional repacker converts variable width codes to a fixed output width during compression or vice versa in decoding. The microprocessor interface provides control and diagnostics of the DCD.

The codec is shown in Fig. 8. The code-word generation block (cw block) is common to both encoding and decoding. The codec has a hierarchical control structure. Each of the blocks has various low-level state machines to make local decisions. The high-level state machine, hilevel sm, monitors the blocks and controls all communications with the outside world under interrupt control. Breaking the control into a hierarchy enabled simplifying the state machines and providing a uniform view of the processing (encoding or decoding) to the outside world. This approach, however, made synchronization of various events more complicated, which was resolved after extensive simulations of the hardware.

Use of the DCD chip requires two external memories to store tables. One memory is 128K×16 b of RAM plus 128K×1 b of resettable memory. The second memory is 64K×31 b of RAM. The DCD has reset and clock inputs, a bidirectional data port, a bidirectional code port, a microprocessor interface, and a direct memory access port for diagnostics. The data port consists of a 15-b bidirectional port and three pipeline control signals. The pipeline control signals are handshake signals to facilitate synchronous transfers to and from the DCD.

During coding, the DCD will accept data at a word transfer rate of approximately 7.5 MHz. During decoding, the rate is higher, about 8 MHz. The actual number of data bits used by the DCD is programmable. The code port consists of a 16-b bidirectional port and three pipeline control signals. The transfer rate at the code port is limited by the transfer rate at the data port and the actual compression ratio achieved. The maximum number of code bits used by the DCD is programmable.

The chip has sixteen 8-b control registers visible to the programmer. All the transfers are under synchronous microprocessor protocol control. The registers control the direction (compress or decompress), the number of data and code bits, and the first allocated code word (allows for keeping a range of reserved code words). They also provide for inserting and removing code words in the code stream. The external code-book memories can be accessed by the DMA of a start address via the control registers (useful for diagnostics). The output code words can be packed into any number of bits (1–16) by the repacker, which is programmed using control registers. The DCD can be programmed to flag an interrupt under various conditions (for example, on receiving a reserved code word or encountering an error). The chip can also be operated in a single-step mode for diagnostics. The DCD can be programmed for automatic reset when the code book is full, or can continue without making any new table entries. The DCD is a standard cell design, implemented in Signetics' 1.5-μm CMOS technology.

The DCD is compatible with existing compression software, *compress* and *lzcomp*, found on UNIX and VMS systems. The speed and compatibility of the DCD makes it useful for compression in disk/tape storage and satellite/microwave/LAN communication systems.

IV. The Compression System

Fig. 9 shows a diagram of an image compression/decompression system (CDS) employing the chip set described above. The configuration shown assumes that the CDS is part of a larger image processing system with a common image system bus for control and image transfer. However, simpler ones are possible. The figure shows only the principal data paths of the CDS. Apart from the SPIPE and DCD chips (including external memories for the code tables), the CDS contains three FIFO's, a local image buffer (LIB), an optional quantizer, and a controller. Two data interfaces accomplish the transfer of pixels and coefficients. We have designed such a system around the VME bus as a two-board set, i.e., SPIPE/LIB board and DCD/quantizer board.

The quantizer is optional and is provided for applications permitting lossy compression. All FIFO's are needed in either mode of the CDS. Two of the FIFO's operate as a multiplexer or demultiplexer of their inputs; the third operates in straight input/output mode. The LIB is large enough to hold the first Σ image of the largest image size the system is designed for, e.g., for a 2K×2K-pixel image, the LIB holds 1 million pixels. The LIB substantially reduces the traffic on the image system bus for hierarchical compression or decompression of large images with many decomposition levels.

In compression, pixels from the original image stream into the CDS and are demultiplexed by the pixel FIFO into two raster streams, i.e., the $[a,b]\cdots$ and $[c,d]\cdots$ lines of the image, subject to pipeline synchronization principles. Then the SPIPE transforms the image. The Σ coefficients are sent to the LIB where they are internally organized as two raster streams—thus the LIB behaves as two FIFO's. Meanwhile, the Δ's are buffered up in the coefficient FIFO. The coefficient FIFO multiplexes the Δ's that pass through the quantizer and the DCD. The output of the DCD is buffered up in the code FIFO and is then transmitted over the image system bus to its final destination. This sequence continues until the entire original image has been processed, i.e., level 1 of the hierarchy has been completed. In this phase, the image system bus is busiest, having to support simultaneously pixel and code transfers. If the Σ subimage is to be further decomposed, i.e., for level 2, the LIB provides the two raster streams $[a,b]\cdots$ and $[c,d]\cdots$, thus the pixel FIFO is inactive. The compression follows the same sequence as before, but now the LIB is both a source and a sink for Σ subimages and the only traffic on the image system bus is that of code transfers. Further hierarchical decomposition steps may be achieved until the smallest image is in the LIB. At this point, we have the option of directly acquiring the smallest image from the LIB through the pixel FIFO or passing it to the DCD for compression.

In decompression, we first load up the smallest image either directly into the LIB through the pixel FIFO or after passing it through the DCD for decompression. To reconstruct the next higher resolution image, the coded Δ's stream into the DCD and are decompressed, demultiplexed by the coefficient FIFO, combined with the smallest image from the LIB, and inverse-transformed by the SPIPE. In this phase, the LIB is both a source and a sink of Σ images, and the only traffic on the image system bus is that of coded Δ's. Once all the Δ's for this layer are processed, the higher resolution image is resident in the LIB. At this point, the image may be directly accessed through the pixel FIFO or decoding of a higher resolution image may be initiated. The reconstruction process proceeds as before until the full-resolution image is achieved. At the final level, the pixels are directly passed through the pixel FIFO to the system bus so that in this phase the image system bus traffic is made up of both coded Δ's and reconstructed pixels.

The above system has been designed around the industry-standard VME bus. For this implementation, the LIB accommodates 1 million pixels, the pixel FIFO consists of two 1K-word buffers, the coefficient FIFO consists of one 2K and 4K word buffers, and the code FIFO of a 1K-word buffer. Several DMA channels are provided to speed up transfers to the image system bus. Although only one DCD

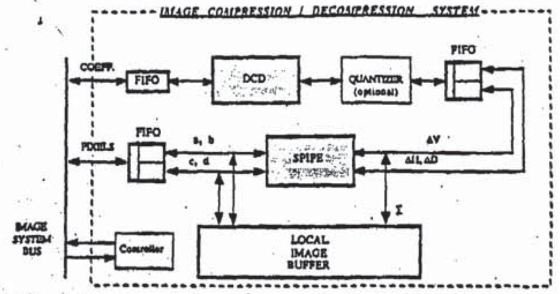


Fig. 9. An image compression decompression system based on the SPIPE and DCD chip set.

device is shown in Fig. 9, it is possible to have up to three if the application requires the Δ's to be separately compressed, as opposed to treating them as a triplet.

V. Results

In [3] extensive simulations were performed for data compression using the *S* transform and Lempel–Ziv coding. The main results are summarized here. The *S* transform generally reduced the entropy only about as effectively as first-order DPCM. For the 15 computed radiograph images used, the zeroth-order entropy was reduced on an average from 8.8 to 4.6 b. This is not surprising, as the filters in the *S* transform are extremely simple, not providing a very good separation between the frequency bands (the Σ and the Δ's). It, however, has the advantage of simple implementation and hierarchical image decomposition.

The above 15 images, each with an original word size of 10 b/pixel, were coded by the L–Z algorithm after being *S*-transformed. An average compression ratio of 2.09 (from 10 b/pixel down to 4.78 b/pixel) was achieved. Thus, for these images, the compression performance was very close to the reduction of the zeroth-order entropy by the *S* transform from an average of 8.8 b/pixel to 4.6 b/pixel.

VI. Conclusion

The described two chips form the basis of a high-speed lossless image compression/decompression system. The *S* transform, besides decorrelating the image, provides a convenient method of hierarchical image decomposition. The data compressor/decompressor IC is a fast and efficient implementation of the L–Z algorithm.

Such a system reduces storage requirements in high-speed image archival and database applications and reduces the transmission time of digital images over communication channels.

Acknowledgment

The authors would like to thank R. Applegate, H. Blume, and T. Wendler for very useful technical discussions, J. Stuyt and G. Hyland for help in various stages of VLSI implementation, and A. Miron for his support.

References

[1] P. Lux, "A novel set of closed orthogonal functions for picture coding," *Arch. Elek. Übertragung.*, vol. 31, pp. 267–274, 1977.
[2] Th. Wendler and D. Meyer-Ebrecht, "Proposed standard for variable format picture processing and a codec approach to match diverse imaging devices," *Proc. SPIE*, vol. 318, pp. 298–305, 1982.
[3] H. Blume and A. Fand, "Reversible and irreversible image data compression using the S-transform and Lempel–Ziv coding," *Proc. SPIE*, vol. 1091, *Medical Imaging III: Image Capture and Display*, pp. 2–18, 1989.
[4] I. A. Shah, O. Akiwumi-Assani, and B. Johnson, "A chip-set for lossless image compression," in *Proc. Custom Integrated Circuits Conf.* (Boston, MA), May 13–16, 1990, paper 17.7.
[5] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Trans. Inform. Theory*, vol. IT-23, pp. 337–343, 1977.
[6] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 530–536, Sept. 1978.
[7] O. Akiwumi-Assani, I. A. Shah, and B. Johnson, "VLSI for picture archiving and communication systems," *Proc. SPIE*, vol. 1246, *Parallel Architectures for Image Processing*, pp. 33–44, Feb. 1990.
[8] S. Ranganath and H. Blume, "Hierarchical image decomposition and filtering using the S transform," *Proc. SPIE*, vol. 914, *Medical Imaging II*, pp. 799–814, 1988.
[9] T. A. Welch, "A technique for high performance data compression," *Computer*, vol. 17, no. 6, pp. 8–19, June 1984.
[10] D. E. Knuth, *The Art Of Computer Programming: Sorting and Searching*, vol. 3. Reading, MA: Addison-Wesley, 1975.
[11] J. S. Vitter, "Analysis of coalesced hashing," Dept. Computer Sci., Stanford Univ., Stanford, CA, Rep. STAN-CS-80-817, 1980.
[12] A. C. Yao, "Uniform hashing is optimal," Dept. Computer Sci., Stanford Univ., Stanford, CA, Rep. STAN-CS-85-1038, 1985.
[13] L. J. Guibas, "The analysis of hashing algorithms," Ph.D. dissertation, Stanford Univ., Stanford, CA, 1976.

Imran Ali Shah was born in Pakistan in 1961. He received the B.S. and M.S. degrees in 1984 and 1986, respectively, from Columbia University, New York, NY. He is currently pursuing the Ph.D. degree, also from Columbia University.

Since 1986, he has been working as a Research Scientist in Philips Research, first in Briarcliff Manor, NY, and now in Eindhoven, The Netherlands. His areas of interest are algorithms and VLSI architectures for digital signal processing and source coding. He is an author of various technical papers, and recipient of U.S. patents in this area.

Mr. Shah is a member of Tau Beta Pi and Eta Kappa Nu.

Olu Akiwumi-Assani received the B.S. degree in physics from the University of Science and Technology, Ghana, in 1973 and the M.S. degree in electrical engineering from Lehigh University, Bethlehem, PA, in 1977. Until 1980 he was enrolled in the Electrical Engineering Ph.D. program at Lehigh.

He taught physics at the University of Science and Technology, Ghana, until he joined Philips Laboratories in 1980. He is now a Senior Member of the Research Staff at Philips Laboratories, Briarcliff Manor, NY. He has been involved in VLSI, computer architecture, and microprocessor systems research activities.

Brian Johnson received the B.S.E.E. degree from Rensselaer Polytechnic Institute, Troy, NY, in 1975 and the M.S.E.E. degree from Syracuse University, Syracuse, NY, in 1978.

He has held positions as Project Leader at Philips' Magnavox CATV and General Electric. He is currently a Project Manager at Philips Laboratories, Briarcliff Manor, NY, where he is involved in VLSI signal processing and systems. He has applied for and received patents in the areas of microprocessor systems, video scrambling, data compression, and digital signal processing.

The JPEG Still Picture Compression Standard

Gregory K. Wallace

Advances over the past decade in many aspects of digital technology—especially devices for image acquisition, data storage, and bitmapped printing and display—have brought about many applications of digital imaging. However, these applications tend to be specialized due to their relatively high cost. With the possible exception of facsimile, digital images are not commonplace in general-purpose computing systems the way text and geometric graphics are. The majority of modern business and consumer usage of photographs and other types of images takes place through more traditional analog means.

The key obstacle for many applications is the vast amount of data required to represent a digital image directly. A digitized version of a single, color picture at TV resolution contains on the order of one million bytes; 35mm resolution requires ten times that amount. Use of digital images often is not viable due to high storage or transmission costs, even when image capture and display devices are quite affordable.

Modern image compression technology offers a possible solution. State-of-the-art techniques can compress typical images from 1/10 to 1/50 their uncompressed size without visibly affecting image quality. But compression technology alone is not sufficient. For digital image applications involving storage or transmission to become widespread in today's marketplace, a *standard* image compression method is needed to enable interoperability of equipment from different manufacturers. The CCITT recommendation for today's ubiquitous Group 3 fax machines [16] is a dramatic example of how a standard compression method can enable an important image application. The Group 3 method, however, deals with bilevel images only and does not address photographic image compression.

For the past few years, a standardization effort known by the acronym JPEG, for Joint Photographic Experts Group, has been working toward establishing the first international digital image compression standard for continuous-tone (multilevel) still images, both grayscale and color. The "joint" in JPEG refers to a collaboration between CCITT and ISO. JPEG convenes officially as the ISO committee designated JTC1/SC2/WG10, but operates in close informal collaboration with CCITT SGVIII.

Photovideotex, desktop publishing, graphic arts, color facsimile, newspaper wirephoto transmission, medical imaging, and many other continuous-tone image applications require a compression standard in order to develop significantly beyond their present state. JPEG has undertaken the ambitious task of developing a *general-purpose* compression standard to meet the needs of almost all continuous-tone still-image applications.

If this goal proves attainable, not only will individual applications flourish, but exchange of images across application boundaries will be facilitated. This latter feature will become increasingly important as more image applications are implemented on general-purpose computing systems, which are themselves becoming increasingly interoperable and internetworked. For applications which require specialized VLSI to meet their compression and decompression speed requirements, a common method will provide economies of scale not possible within a single application.

This article gives an overview of JPEG's proposed image-compression standard. Readers without prior knowledge of JPEG or compression based on the Discrete Cosine Transform (DCT) are encouraged to study first the detailed description of the Baseline sequential codec, which is the basis for all of the DCT-based decoders. While this article provides many details, many more are necessarily omitted. The reader should refer to the ISO draft standard [2] before attempting implementation.

Interestingly, some of the earliest industry attention to the JPEG proposal has been focused on the Baseline sequential codec as a *motion* (intraframe) image compression method. (See the associated sidebar, "NEXTstep: Putting JPEG to Multiple Uses.") The fact that it has not been in JPEG's charter as an ISO working group to address this application may indicate that distinction between still- and motion-image coding can sometimes be artificial.

Background: Requirements and Selection Process

JPEG's goal has been to develop a method for continuous-tone image compression which meets the following requirements:

a) be at or near the state of the art with regard to compression rate and accompanying image fidelity, over a wide range of image quality ratings, and especially in the range where visual fidelity to the original is characterized as "very good" to "excellent"; also, the encoder should be parameterizable, so that the application (or user) can set the desired compression/quality tradeoff;

b) be applicable to practically any kind of continuous-tone digital source image (i.e., for most practical purposes not be restricted to images of certain dimensions, color spaces, pixel aspect ratios, etc.), and not be limited to classes of imagery with restrictions on scene content, such as complexity, range of colors, or statistical properties;

c) have tractable computational complexity, to make feasible software implementations with viable performance on a range of CPUs, as well as hardware implementations with viable cost for applications requiring high performance;

d) have the following modes of operation:

- Sequential encoding: each image component is encoded in a single left-to-right, top-to-bottom scan;
- Progressive encoding: the image is encoded in multiple scans for applications in which transmission time is long, and the viewer prefers to watch the image build up in multiple coarse-to-clear passes;
- Lossless encoding: the image is encoded to guarantee exact recovery of every source image sample value (even though the result is low compression compared to the lossy modes);
- Hierarchical encoding: the image is encoded at multiple resolutions, so that lower-resolution versions may be accessed without first having to decompress the image at its full resolution.

In June 1987, JPEG conducted a selection process based on a blind assessment of subjective picture quality, and narrowed 12 proposed methods to three. Three informal working groups formed to refine them, and in January 1988, a second, more rigorous selection process [18] revealed the "ADCT" proposal [10], based on the 8 × 8 DCT, had produced the best picture quality.

At the time of its selection, the DCT-based method was only partially defined for some of the modes of operation. From 1988 through 1990, JPEG undertook the sizable task of defining, documenting, simulating, testing, validating, and simply agreeing on the plethora of details necessary for genuine interoperability and universality. Further history of the JPEG effort is contained in [5, 6, 8, 17].

Architecture of the Proposed Standard

The proposed standard contains the four "modes of operation" identified previously. For each mode, one or more distinct codecs are specified. Codecs within a mode differ according to the precision of source image samples they can handle or the entropy coding method they use. Although the word codec (encoder/decoder) is used frequently in this article, there is no requirement that implementations must include both an encoder and a decoder. Many applications will have systems or devices which require only one or the other.

The four modes of operation and their various codecs have resulted from JPEG's goal of being generic and from the diversity of image formats across applications. The multiple pieces can give the impression of undesirable complexity, but they should actually be regarded as a comprehensive "toolkit" which can span a wide range of continuous-tone image applications. It is unlikely that many implementations will utilize every tool—indeed, most of the early implementations now on the market (even before final ISO approval) have implemented only the Baseline sequential codec.

The Baseline sequential codec is inherently a rich and sophisticated compression method which will be sufficient for many applications. Getting just this minimum JPEG capability implemented properly and interoperably will provide the industry with an important initial capability for exchange of images across vendors and applications.

Processing Steps for DCT-Based Coding

Figures 1 and 2 show the key processing steps which are the heart of the DCT-based modes of operation. These figures illustrate the special case of single-component (grayscale) image compression. The reader can grasp the essentials of DCT-based compression by thinking of it as essentially compression of a stream of 8 × 8 blocks of grayscale image samples. Color image compression can then be approximately regarded as compression of multiple grayscale images, which are either compressed entirely one at a time, or are compressed by alternately interleaving 8 × 8 sample blocks from each in turn.

For DCT sequential-mode codecs, which include the Baseline sequential codec, the simplified diagrams indicate how single-component compression works in a fairly complete way. Each 8 × 8 block is input, makes its way through each processing step, and yields output in compressed form into the data stream. For DCT progressive-mode codecs, an image buffer exists prior to the entropy coding step, so that an image can be stored and then parcelled out in multiple scans with successively improving quality. For the hierarchical mode of operation, the steps shown are used as building blocks within a larger framework.

8 × 8 FDCT and IDCT

At the input to the encoder, source image samples are grouped into 8 × 8 blocks, shifted from unsigned integers with range $[0, 2^P - 1]$ to signed integers with range $[-2^{P-1}, 2^{P-1} - 1]$, and input to the Forward DCT (FDCT). At the output from the decoder, the Inverse DCT (IDCT) outputs 8 × 8 sample blocks to form the reconstructed image. The following equations are the idealized mathematical definitions of the 8 × 8 FDCT and 8 × 8 IDCT:

$$F(u,v) = \frac{1}{4}C(u)C(v)\left[\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y) * \cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16}\right] \quad (1)$$

$$f(x,y) = \frac{1}{4}\left[\sum_{u=0}^{7}\sum_{v=0}^{7} C(u)C(v)F(u,v) * \cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16}\right]$$

where: $C(u), C(v) = 1/\sqrt{2}$ for $u, v = 0$; $C(u), C(v) = 1$ otherwise. (2)

The DCT is related to the Discrete Fourier Transform (DFT). Some simple intuition for DCT-based compression can be obtained by viewing the FDCT as a harmonic analyzer and the IDCT as a harmonic synthesizer. Each 8 × 8 block of source image samples is effectively a 64-point discrete signal which is a function of the two spatial dimensions *x* and *y*. The FDCT takes such a signal as its input and decomposes it into 64 orthogonal basis signals. Each contains one of the 64 unique two-dimensional (2D) "spatial frequencies" which comprise the input signal's "spectrum." The output of the FDCT is the set of 64 basis-signal amplitudes or "DCT coefficients" whose values are uniquely determined by the particular 64-point input signal.

The DCT coefficient values can thus be regarded as the relative amounts of the 2D spatial frequencies contained in the 64-point input signal. The coefficient with zero frequency in both dimensions is called the "DC coefficient" and the remaining 63 coefficients are called the "AC coefficients." Because sample values typically vary slowly from point to point across an image, the FDCT processing step lays the foundation for achieving data compression by concentrating most of the signal in the lower spatial frequencies. For a typical 8 × 8 sample block from a typical source image, most of the spatial frequencies have zero or near-zero amplitude and need not be encoded.

At the decoder the IDCT reverses this processing step. It takes the 64 DCT coefficients (which at that point have been quantized) and reconstructs a 64-point output image signal by summing the basis signals. Mathematically, the DCT is a one-to-one mapping of 64-point vectors between the image and the frequency domains. If the FDCT and IDCT could be computed with perfect accuracy and if the DCT coefficients were not quantized as in the following description, the original 64-point signal could be exactly recovered. In principle, the DCT introduces no loss to the source image samples; it merely transforms them to a domain in which they can be more efficiently encoded.

Some properties of practical FDCT and IDCT implementations raise the issue of what precisely should be required by the JPEG standard. A fundamental property is that the FDCT and IDCT equations contain transcendental functions. Consequently, no physical implementation can compute them with perfect accuracy. Because of the DCT's application importance and its relationship to the DFT, many different algorithms by which the FDCT and IDCT may be approximately computed have been devised [15]. Indeed, research in fast DCT algorithms is ongoing,

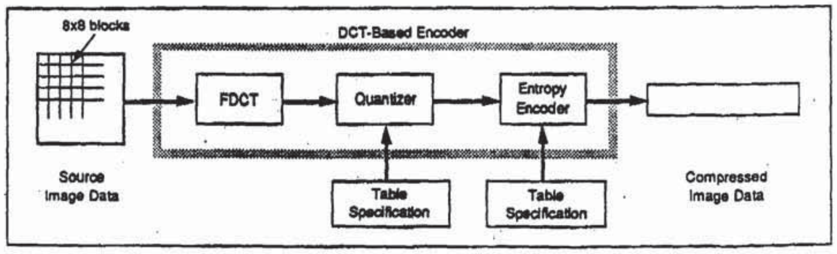


FIGURE 1.
DCT-Based Encoder Processing Steps

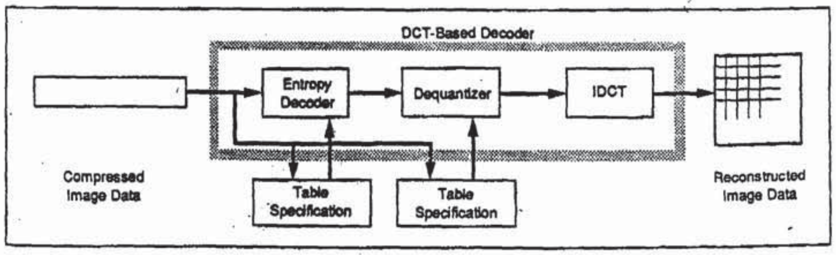


FIGURE 2.
DCT-Based Decoder Processing Steps

and no single algorithm is optimal for all implementations. What is optimal in software for a general-purpose CPU is unlikely to be optimal in firmware for a programmable DSP and is certain to be suboptimal for dedicated VLSI.

Even in light of the finite precision of the DCT inputs and outputs, independently designed implementations of the very same FDCT or IDCT algorithm which differ even minutely in the precision by which they represent cosine terms or intermediate results, or in the way they sum and round fractional values, will eventually produce slightly different outputs from identical inputs.

To preserve freedom for innovation and customization within implementations, JPEG has chosen to specify neither a unique FDCT algorithm nor a unique IDCT algorithm in its proposed standard. This makes compliance somewhat more difficult to confirm, because two compliant encoders (or decoders) generally will not produce identical outputs given identical inputs. The JPEG standard will address this issue by specifying an accuracy test as part of its compliance tests for all DCT-based encoders and decoders; this is to ensure against crudely inaccurate cosine basis functions which would degrade image quality.

For each DCT-based mode of operation, the JPEG proposal specifies separate codecs for images with 8-bit and 12-bit (per component) source image samples. The 12-bit codecs, needed to accommodate certain types of medical and other images, require greater computational resources to achieve the required FDCT or IDCT accuracy. Images with other sample precisions can usually be accommodated by either an 8-bit or 12-bit codec, but this must be done outside the JPEG standard. It is the responsibility of applications to decide how to fit or pad a 6-bit sample into the 8-bit encoder's input interface, how to unpack it at the decoder's output, and how to encode any necessary related information.

Quantization

After output from the FDCT, each of the 64 DCT coefficients is uniformly quantized in conjunction with a 64-element Quantization Table, which must be specified by the application (or user) as an input to the encoder. Each element can be any integer value from 1 to 255, which specifies the step size of the quantizer for its corresponding DCT coefficient. The purpose of quantization is to achieve further compression by representing DCT coefficients with no greater precision than is necessary to achieve the desired image quality. Stated another way, the goal of this processing step is to discard information which is not visually significant. Quantization is a many-to-one mapping, and therefore is fundamentally lossy. It is the principal source of lossiness in DCT-based encoders.

Quantization is defined as division of each DCT coefficient by its corresponding quantizer step size, followed by rounding to the nearest integer:

$$F^Q(u,v) = \mathit{Integer\ Round}\left(\frac{F(u,v)}{Q(u,v)}\right). \tag{3}$$

This output value is normalized by the quantizer step size. Dequantization is the inverse function, which in this case means simply that the normalization is removed by multiplying by the step size, which returns the result to a representation appropriate for input to the IDCT:

$$F^{Q'}(u,v) = F^Q(u,v) * Q(u,v) \tag{4}$$

When the aim is to compress the image as much as possible without visible artifacts, each step size ideally should be chosen as the perceptual threshold or "just noticeable difference" for the visual contribution of its corresponding cosine basis function. These thresholds are also functions of the source image characteristics, display characteristics and viewing distance. For applications in which these variables can be reasonably well defined, psychovisual experiments can be performed to determine the best thresholds. The experiment described in [11] has led to a set of Quantization Tables for CCIR-601 [4] images and displays. These have been used experimentally by JPEG members and will appear in the ISO standard as a matter of information, but not as a requirement.

DC Coding and Zig-Zag Sequence

After quantization, the DC coefficient is treated separately from the 63 AC coefficients. The DC coefficient is a measure of the average value of the 64 image samples. Because there is usually strong correlation between the DC coefficients of adjacent 8 × 8 blocks, the quantized DC coefficient is encoded as the difference from the DC term of the previous block in the encoding order (defined in the following), as shown in Figure 3. This special treatment is worthwhile, as DC coefficients frequently contain a significant fraction of the total image energy.

Finally, all of the quantized coefficients are ordered into the "zig-zag" sequence, also shown in Figure 3. This ordering helps to facilitate entropy coding by placing low-frequency coefficients (which are more likely to be nonzero) before high-frequency coefficients.

Entropy Coding

The final DCT-based encoder processing step is entropy coding. This step achieves additional compression losslessly by encoding the quantized DCT coefficients more compactly based on their statistical characteristics. The JPEG proposal specifies two entropy coding methods—Huffman coding [7] and arithmetic coding [14]. The Baseline sequential codec uses Huffman coding, but codecs with both methods are specified for all modes of operation.

It is useful to consider entropy

coding as a 2-step process. The first step converts the zig-zag sequence of quantized coefficients into an intermediate sequence of symbols. The second step converts the symbols to a data stream in which the symbols no longer have externally identifiable boundaries. The form and definition of the intermediate symbols is dependent on both the DCT-based mode of operation and the entropy coding method.

Huffman coding requires that one or more sets of Huffman code tables be specified by the application. The same tables used to compress an image are needed to decompress it. Huffman tables may be predefined and used within an application as defaults, or computed specifically for a given image in an initial statistics-gathering pass prior to compression. Such choices are the business of the applications which use JPEG; the JPEG proposal specifies no required Huffman tables. Huffman coding for the Baseline sequential encoder is described in detail later in this article.

By contrast, the particular arithmetic coding method specified in the JPEG proposal [2] requires no tables to be externally input, because it is able to adapt to the image statistics as it encodes the image. (If desired, statistical conditioning tables can be used as inputs for slightly better efficiency, but this is not required.) Arithmetic coding has produced 5–10% better compression than Huffman for many of the images which JPEG members have tested. However, some feel it is more complex than Huffman coding for certain implementations, for example, the highest-speed hardware implementations. (Throughout JPEG's history, "complexity" has proved to be most elusive as a practical metric for comparing compression methods.)

If the only difference between two JPEG codecs is the entropy coding method, transcoding between the two is possible by simply entropy decoding with one method and entropy recoding with the other.

Compression and Picture Quality

For color images with moderately complex scenes, all DCT-based modes of operation typically produce the following levels of picture quality for the indicated ranges of compression. These levels are only a guideline—quality and compression can vary significantly according to source image characteristics and scene content. (The units "bits/pixel" here mean the total number of bits in the compressed image—including the chrominance components—divided by the number of samples in the luminance component.)

- 0.25–0.5 bits/pixel: moderate to good quality, sufficient for some applications;
- 0.5–0.75 bits/pixel: good to very good quality, sufficient for many applications;
- 0.75–1.5 bits/pixel: excellent quality, sufficient for most applications;
- 1.5–2.0 bits/pixel: usually indistinguishable from the original, sufficient for the most demanding applications.

Later in this article, Figure 11 shows an example of the picture quality obtained for a CCIR-601 image at various stages and bit rates of a progressive encoding. Because FDCT and Quantization are common to progressive and sequential DCT-based modes, the quality and compression shown in Figure 11 is also indicative of the trade-offs that can be expected for sequential coding.

Processing Steps for Predictive Lossless Coding

After its selection of a DCT-based method in 1988, JPEG discovered that a DCT-based lossless mode was difficult to define as a practical standard against which encoders and decoders could be independently implemented, without placing severe constraints on both encoder and decoder implementations.

JPEG, to meet its requirement for a lossless mode of operation, has chosen a simple predictive method which is wholly independent of the DCT processing described previously. Although not the result of rigorous competitive evaluation as was the DCT-based method, the predictive method produces results which, in light of its simplicity, are surprisingly close to the state of the art for lossless continuous-tone compression.

Figure 4 shows the main processing steps for a single-component image. A predictor combines the

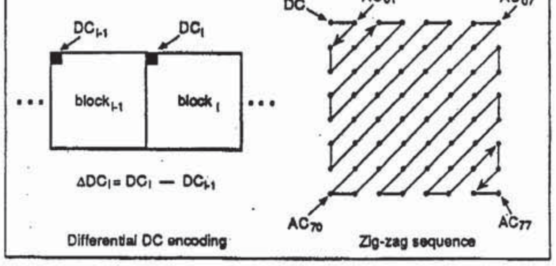


FIGURE 3.
Preparation of Quantized Coefficients for Entropy Coding

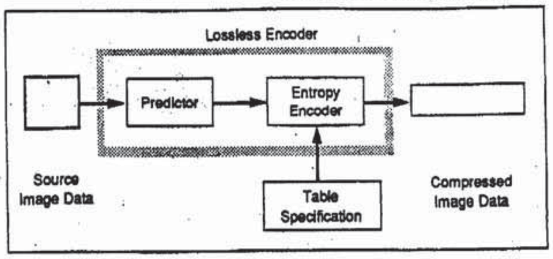


FIGURE 4.
Lossless Mode Encoder Processing Steps

TABLE 1.

Predictors for-Lossless Coding

SELECTION-VALUE	PREDICTION
0	no prediction
1	A
2	B
3	C
4	A + B − C
5	A + ((B − C)/2)
6	B + ((A − C)/2)
7	(A + B)/2

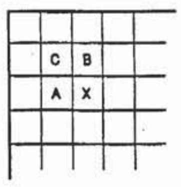


FIGURE 5.
3-Sample Prediction Neighborhood

values of up to three neighboring samples (A, B, and C) to form a prediction of the sample indicated by X in Figure 5. This prediction is then subtracted from the actual value of sample X, and the difference is encoded losslessly by either of the entropy coding methods—Huffman or arithmetic. Any one of the eight predictors listed in Table 1 (under "selection-value") can be used.

Selections 1, 2 and 3 are one-dimensional predictors and selections 4, 5, 6, and 7 are two-dimensional predictors. Selection-value 0 can only be used for differential coding in the hierarchical mode of operation. The entropy coding is nearly identical to that used for the DC coefficient as described later (for Huffman coding).

For the lossless mode of operation, two different codecs are specified—one for each entropy coding method. The encoders can use any source image precision from 2 to 16 bits/sample, and can use any of the predictors except selection-value 0. The decoders must handle any of the sample precisions and any of the predictors.

Lossless codecs typically produce around 2:1 compression for color images with moderately complex scenes.

Multiple-Component Images

The previous sections discussed the key processing steps of the DCT-based and predictive lossless codecs for the case of single-component source images. These steps accomplish the image data compression. But a good deal of the JPEG proposal is also concerned with the handling and control of color (or other) images with multiple components. JPEG's aim for a generic compression standard requires its proposal to accommodate a variety of source image formats.

Source Image Formats

The source image model used in the JPEG proposal is an abstraction from a variety of image types and applications, and consists only of what is necessary to compress and reconstruct digital image data. The reader should recognize that the JPEG compressed data format does not encode enough information to serve as a complete image representation. For example, JPEG does not specify or encode any information on pixel aspect ratio, color space, or image acquisition characteristics.

Figure 6 illustrates the JPEG source image model. A source image contains from 1 to 255 image components, sometimes called color or spectral bands or channels. Each component consists of a rectangular array of samples. A sample is defined to be an unsigned integer with precision P bits, with any value in the range $[0, 2^P - 1]$. All samples of all components within the same source image must have the same precision P. P can be 8 or 12 for DCT-based codecs, and 2 to 16 for predictive codecs.

The ith component has sample dimensions x_i by y_i. To accommodate formats in which some image components are sampled at different rates than others, components can have different dimensions. The dimensions must have a mutual integral relationship defined by H_i and V_i, the relative horizontal and vertical sampling factors, which must be specified for each component. Overall image dimensions X and Y are defined as the maximum x_i and y_i for all components in the image, and can be any number up to 2^{16}. H and V are allowed only the integer values 1 through 4. The encoded parameters are X, Y, and the H_is and V_is for each component. The decoder reconstructs the dimensions x_i and y_i for each component, according to the following relationship shown in Equation 5:

$$x_i = \left\lceil X \times \frac{H_i}{H_{max}} \right\rceil \text{ and } \quad y_i = \left\lceil Y \times \frac{V_i}{V_{max}} \right\rceil \qquad (5)$$

where ⌈ ⌉ is the ceiling function.

Encoding Order and Interleaving

A practical image compression standard must address how systems will need to handle the data during the process of decompression. Many applications need to pipeline the process of displaying or printing multiple-component images in parallel with the process of decompression. For many systems, this is only feasible if the components are interleaved together within the compressed data stream.

To make the same interleaving machinery applicable to both DCT-based and predictive codecs, the JPEG proposal has defined the concept of "data unit." A data unit is a sample in predictive codecs and an 8 × 8 block of samples in DCT-based codecs.

The order in which compressed data units are placed in the compressed data stream is a generalization of raster-scan order. Generally, data units are ordered from left-to-right and top-to-bottom according to the orientation shown in Figure 6. (It is the responsibility of applications to define which edges of a source image are top, bottom, left, and right.) If an image component is noninterleaved (i.e., compressed without being interleaved with other components), compressed data units are ordered in a pure raster scan as shown in Figure 7.

When two or more components are interleaved, each component C_i is partitioned into rectangular regions of H_i by V_i data units, as shown in the generalized example of Figure 8. Regions are ordered within a component from left-to-right and top-to-bottom, and within a region, data units are ordered from left-to-right and top-to-bottom. The JPEG proposal defines the term Minimum Coded Unit (MCU) to be the smallest group of interleaved data units. For the example shown, MCU_1 consists of data units taken first from the top-left-most region of C_1, followed by data units from the same region of C_2, and likewise for C_3 and C_4. MCU_2 continues the pattern as shown.

Thus, interleaved data is an ordered sequence of MCUs, and the number of data units contained in an MCU is determined by the number of components interleaved and their relative sampling factors. The maximum number of components which can be interleaved is 4 and the maximum number of data units in an MCU is 10. The latter restriction is expressed as shown in Equation 6, where the summation is over the interleaved components:

$$\sum_{\substack{all\ i\ in\\ interleave}} H_i \times V_i \leq 10 \qquad (6)$$

Because of this restriction, not every combination of 4 components which can be represented in noninterleaved order within a JPEG-compressed image is allowed to be interleaved. Also, note that the JPEG proposal allows some components to be interleaved and some to be noninterleaved within the same compressed image.

Multiple Tables

In addition to the interleaving control discussed previously, JPEG codecs must control application of the proper table data to the proper components. The same quantization table and the same entropy coding table (or set of tables) must be used to encode all samples within a component.

JPEG decoders can store up to 4 different quantization tables and up to 4 different (sets of) entropy coding tables simultaneously. (The Baseline sequential decoder is the exception; it can only store up to 2 sets of entropy coding tables.) This is necessary for switching between different tables during decompression of a scan containing multiple (interleaved) components, in order to apply the proper table to the proper component. (Tables cannot be loaded during decompression of a scan.) Figure 9 illustrates the table-switching control that must be managed in conjunc-

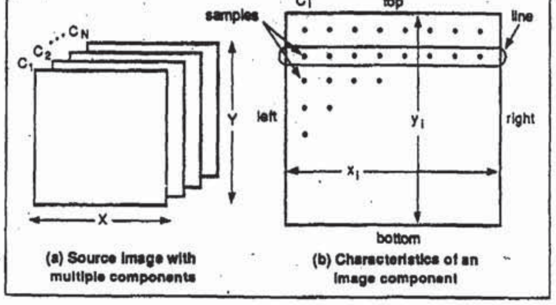

samples
C_1
C_2
C_N
Y
X
C_i
top
line
left
y_i
right
x_i
bottom
(a) Source image with multiple components
(b) Characteristics of an image component

FIGURE 6.
JPEG Source Image Model

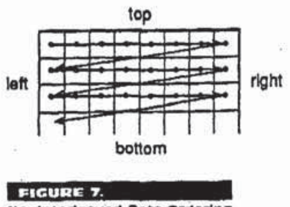
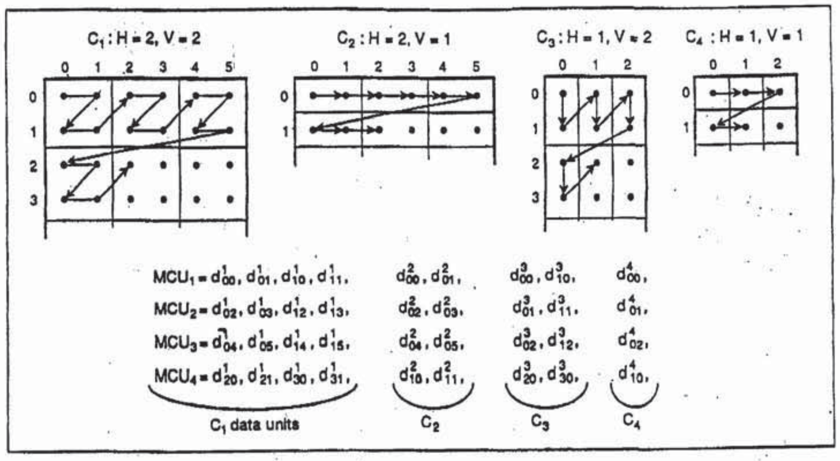


FIGURE 7.
Noninterleaved Data Ordering



$$\begin{aligned}
MCU_1 &= \underbrace{d^1_{00}, d^1_{01}, d^1_{10}, d^1_{11},}\ d^2_{00}, d^2_{01},\ d^3_{00}, d^3_{10},\ d^4_{00},\\
MCU_2 &= d^1_{02}, d^1_{03}, d^1_{12}, d^1_{13},\ d^2_{02}, d^2_{03},\ d^3_{01}, d^3_{11},\ d^4_{01},\\
MCU_3 &= d^1_{04}, d^1_{05}, d^1_{14}, d^1_{15},\ d^2_{04}, d^2_{05},\ d^3_{02}, d^3_{12},\ d^4_{02},\\
MCU_4 &= d^1_{20}, d^1_{21}, d^1_{30}, d^1_{31},\ d^2_{10}, d^2_{11},\ d^3_{20}, d^3_{30},\ d^4_{10},
\end{aligned}$$

FIGURE 8.
Generalized Interleaved Data Ordering Example

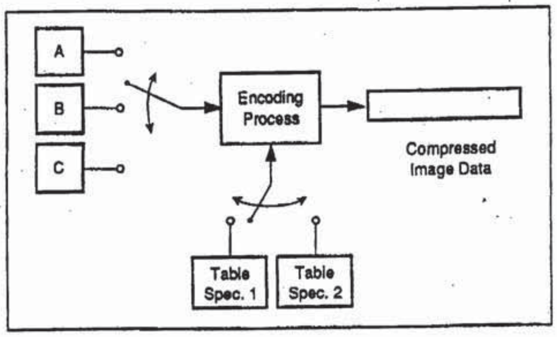


FIGURE 9.
Component-Interleave and Table-Switching Control

tion with multiple-component interleaving for the encoder side. (This simplified view does not distinguish between quantization and entropy coding tables.)

Baseline and Other DCT Sequential Codecs

The DCT sequential mode of operation consists of the FDCT and Quantization steps from the section entitled "Processing Steps for DCT-Based Coding" and the multiple-component control from the previous section on multiple component images. In addition to the Baseline sequential codec, other DCT sequential codecs are defined to accommodate the two different sample precisions (8 and 12 bits) and the two different types of entropy coding methods (Huffman and arithmetic).

Baseline sequential coding is for images with 8-bit samples and uses Huffman coding only. It also differs from the other sequential DCT codecs in that its decoder can store only two sets of Huffman tables (one AC table and one DC table per set). This restriction means that, for images with three or four interleaved components, at least one set of Huffman tables must be shared by two components. This restriction poses no limitation at all for noninterleaved components; a new set of tables can be loaded into the decoder before decompression of a noninterleaved component begins.

For many applications which do need to interleave three color components, this restriction is hardly a limitation at all. Color spaces (YUV, CIELUV, CIELAB, and others) which represent the chromatic ("color") information in two components and the achromatic ("grayscale") information in a third are more efficient for compression than spaces like RGB. One Huffman table set can be used for the achromatic component and one for the chrominance components. DCT coefficient statistics are similar for the chrominance components of most images, and one set of Huffman tables can encode both almost as optimally as two.

The committee also felt that early availability of single-chip implementations at commodity prices would encourage early acceptance of the JPEG proposal in a variety of applications. In 1988 when Baseline sequential was defined, the committee's VLSI experts felt that current technology made the feasibility of crowding four sets of loadable Huffman tables—in addition to four sets of Quantization tables—onto a single commodity-priced codec chip a risky proposition.

The FDCT, Quantization, DC differencing, and zig-zag ordering processing steps for the Baseline sequential codec proceed just as described in the section "Processing Steps for DCT-Based Coding." Prior to entropy coding, there usually are few nonzero and many zero-valued AC coefficients. The task of entropy coding is to encode these few coefficients efficiently. The description of Baseline sequential entropy coding is given in two steps: conversion of the quantized DCT coefficients into an intermediate sequence of symbols and assignment of variable-length codes to the symbols.

Intermediate Entropy Coding Representations

In the intermediate symbol sequence, each nonzero AC coefficient is represented in combination with the "runlength" (consecutive number) of zero-valued AC coefficients which precede it in the zig-zag sequence. Each such runlength/nonzero-coefficient combination is (usually) represented by a pair of symbols:

symbol-1
(RUNLENGTH, SIZE)

symbol-2
(AMPLITUDE)

Symbol-1 represents two pieces of information, RUNLENGTH and SIZE. Symbol-2 represents the single piece of information designated AMPLITUDE, which is simply the amplitude of the nonzero AC coefficient. RUNLENGTH is the number of consecutive zero-valued AC coefficients in the zig-zag sequence preceding the nonzero AC coefficient being represented. SIZE is the number of bits used to encode AMPLITUDE—that is, to encode symbol-2, by the signed-integer encoding used with JPEG's particular method of Huffman coding.

RUNLENGTH represents zero-runs of length 0 to 15. Actual zero-runs in the zig-zag sequence can be greater than 15, so the symbol-1 value (15, 0) is interpreted as the extension symbol with runlength = 16. There can be up to three consecutive (15, 0) extensions before the terminating symbol-1 whose RUNLENGTH value completes the actual runlength. The terminating symbol-1 is always followed by a single symbol-2, except for the case in which the last run of zeros includes the last (63d) AC coefficient. In this frequent case, the special symbol-1 value (0, 0) means EOB (end of block), and can be viewed as an "escape" symbol which terminates the 8 × 8 sample block.

Thus, for each 8 × 8 block of samples, the zig-zag sequence of 63 quantized AC coefficients is represented as a sequence of symbol-1, symbol-2 symbol-pairs, though each "pair" can have repetitions of symbol-1 in the case of a long runlength or only one symbol-1 in the case of an EOB.

The possible range of quantized AC coefficients determines the range of values which both the AMPLITUDE and the SIZE information must represent. A numerical analysis of the 8 × 8 FDCT equation shows that, if the 64-point (8 × 8 block) input signal contains N-bit integers, then the nonfractional part of the output numbers (DCT coefficients) can grow by at most 3 bits. This is also the largest possible size of a quantized DCT coefficient when its quantizer step size has integer value 1.

Baseline sequential has 8-bit integer source samples in the range $[2^7, 2^7 - 1]$, so quantized AC coefficient amplitudes are covered by integers in the range $[-2^{10}, 2^{10} - 1]$. The signed-integer encoding uses symbol-2 AMPLITUDE codes of 1 to 10 bits in length (so SIZE also represents values from 1 to 10), and RUNLENGTH represents values from 0 to 15 as discussed previously. For AC coefficients, the structure of the symbol-1 and symbol-2 intermediate representations is illustrated in Tables 2 and 3, respectively.

The intermediate representation for an 8 × 8 sample block's differential DC coefficient is structured similarly. Symbol-1, however, represents only SIZE information; symbol-2 represents AMPLITUDE information as before:

symbol-1 , symbol-2
(SIZE) (AMPLITUDE)

Because the DC coefficient is differentially encoded, it is covered by twice as many integer values, $[-2^{11}, 2^{11} - 1]$ as the AC coefficients, so

TABLE 2.

Baseline Huffman Coding Symbol-1 Structure

		SIZE					
		0	1	2	. . .	9	10
RUNLENGTH	0	EOB					
	.	X					
	.	X			RUN-SIZE VALUES		
	.	X					
	15	ZRL					

one additional level must be added to the bottom of Table 3 for DC coefficients. Symbol-1 for DC coefficients thus represents a value from 1 to 11.

Variable-Length Entropy Coding

Once the quantized coefficient data for an 8 × 8 block is represented in the intermediate symbol sequence described above, variable-length codes are assigned. For each 8 × 8 block, the DC coefficient's symbol-1 and symbol-2 representation is coded and output first.

For both DC and AC coefficients, each symbol-1 is encoded with a variable-length code (VLC) from the Huffman table set assigned to the 8 × 8 block's image component. Each symbol-2 is encoded with a "variable-length integer" (VLI) code whose length in bits is given in Table 3. VLCs and VLIs both are codes with variable lengths, but VLIs are not Huffman codes. An important distinction is that the length of a VLC (Huffman code) is not known until it is decoded, but the length of a VLI is stored in its preceding VLC.

Huffman codes (VLCs) must be specified externally as an input to JPEG encoders. (Note that the form in which Huffman tables are represented in the data stream is an indirect specification with which the decoder must construct the tables themselves prior to decompression.) The JPEG proposal includes an example set of Huffman tables in its informational annex, but because they are application-specific, it specifies none for required use. The VLI codes, in contrast, are "hardwired" into the proposal. This is appropriate, because the VLI codes are far more numerous, can be computed rather than stored, and have not been shown to be appreciably more efficient when implemented as Huffman codes.

Other DCT Sequential Codecs

The structure of the 12-bit DCT sequential codec with Huffman coding is a straightforward extension of the entropy coding method described previously. Quantized DCT coefficients can be 4 bits larger, so the SIZE and AMPLITUDE information extend accordingly. DCT sequential with arithmetic coding is described in detail in [2].

DCT Progressive Mode

The DCT progressive mode of operation consists of the same FDCT and Quantization steps from the section "Processing Steps for DCT-Based Coding" that are used by DCT sequential mode. The key difference is that each image component is encoded in multiple scans rather than in a single scan. The first scan(s) encode a rough but recognizable version of the image which can be transmitted quickly in comparison to the total transmission time, and are refined by succeeding scans until reaching the level of picture quality that was established by the quantization tables.

To achieve this requires the addition of an image-sized buffer memory at the output of the quantizer, before the input to the entropy encoder. The buffer memory must be of sufficient size to store the image as quantized DCT coefficients, each of which (if stored straightforwardly) is 3 bits larger than the source image samples. After each block of DCT coefficients is quantized, it is stored in the coefficient buffer memory. The buffered coefficients are then partially encoded in each of multiple scans.

TABLE 3.

Baseline Entropy Coding Symbol-2 Structure

SIZE	AMPLITUDE
1	−1,1
2	−3,−2,2,3
3	−7..−4,4..7
4	−15..−8,8..15
5	−31..−16,16..31
6	−63..−32,32..63
7	−127..−64,64..127
8	−255..−128,128..255
9	−511..−256,256..511
10	−1023..−512,512..1023

There are two complementary methods by which a block of quantized DCT coefficients may be partially encoded. First, only a specified "band" of coefficients from the zig-zag sequence need be encoded within a given scan. This procedure is called "spectral selection," because each band typically contains coefficients which occupy a lower or higher part of the spatial-frequency spectrum for that 8 × 8 block. Secondly, the coefficients within the current band need not be encoded to their full (quantized) accuracy in a given scan. Upon a coefficient's first encoding, the N most significant bits can be encoded first, where N is specifiable. In subsequent scans, the less significant bits can then be encoded. This procedure is called successive approximation. Both procedures can be used separately, or mixed in flexible combinations.

Some intuition for spectral selection and successive approximation can be obtained from Figure 10. The quantized DCT coefficient information can be viewed as a rectangle for which the axes are the DCT coefficients and their amplitudes. Spectral selection slices the information in one dimension and successive approximation in the other.

For comparative purposes, Figure 11 shows an example of both progressive encoding methods.

Hierarchical Mode Of Operation

The hierarchical mode provides a "pyramidal" encoding of an image at multiple resolutions, each differing in resolution from its adjacent encoding by a factor of two in either the horizontal or vertical dimension or both. The encoding procedure can be summarized as follows:

a) Filter and down-sample the original image by the desired number of multiples of 2 in each dimension.

b) Encode this reduced-size image using one of the sequential DCT, progressive DCT, or lossless encoders described previously.

c) Decode this reduced-size image and then interpolate and up-sample it by 2 horizontally and/or vertically, using the identical interpolation filter which the receiver must use.

d) Use this up-sampled image as a prediction of the original at this resolution, and encode the difference image using one of the sequential DCT, progressive DCT, or lossless encoders described previously.

e) Repeat steps c) and d) until the full resolution of the image has been encoded.

The encoding in steps b) and d) may be done using only DCT-based processes, only lossless processes, or DCT-based processes with a final lossless process for each component.

Hierarchical encoding is useful in applications in which a very high resolution image must be accessed by a lower-resolution device, which does not have the buffer capacity to reconstruct the image at its full resolution and then scale it down for the lower-resolution display. An example is an image scanned and compressed at high resolution for a very high-quality printer, where the image must also be displayed on a low-resolution PC video screen.

Other Aspects of the JPEG Proposal

Some key aspects of the proposed standard can only be mentioned briefly. Foremost among these are points concerning the coded representation for compressed image data specified in addition to the encoding and decoding procedures.

Most importantly, an *interchange format* syntax is specified which ensures that a JPEG-compressed image can be exchanged successfully between different application environments. The format is structured in a consistent way for all modes of operation. The interchange format always includes all quantization and entropy-coding

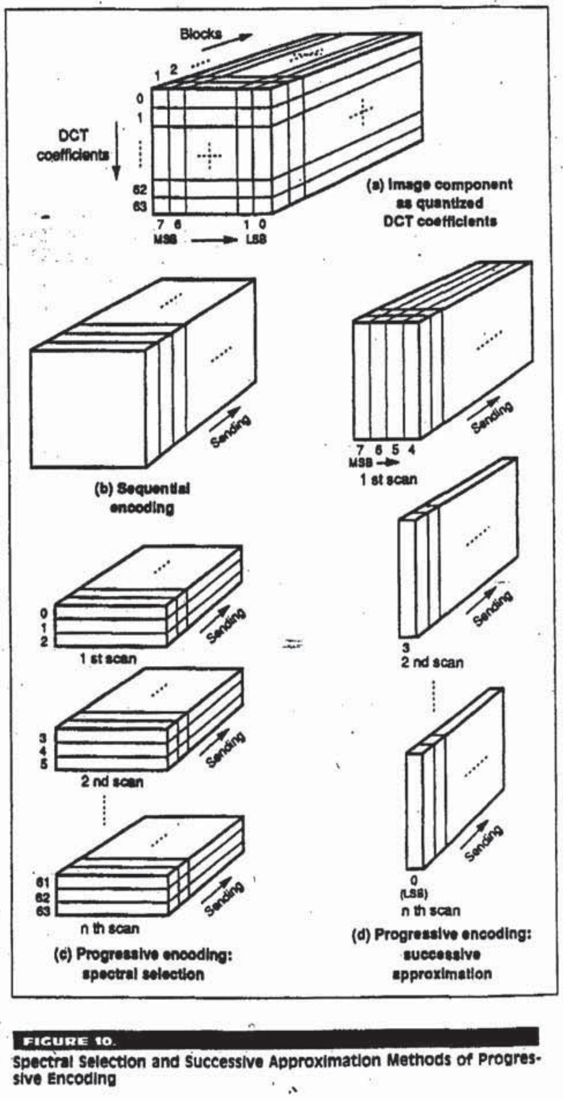


FIGURE 10.
Spectral Selection and Successive Approximation Methods of Progressive Encoding

tables which were used to compress the image.

Applications (and application-specific standards) are the "users" of the JPEG standard. The JPEG standard imposes no requirement that, within an application's environment, all or even any tables must be encoded with the compressed image data during storage or transmission. This leaves applications the freedom to specify default or referenced tables if they are considered appropriate. It also leaves them the responsibility to ensure that JPEG-compliant decoders used within their environment get loaded with the proper tables at the proper times, and that the proper tables are included in the interchange format when a compressed image is "exported" outside the application.

Some of the important applications that are already in the process of adopting JPEG compression or have stated their interest in doing so are Adobe's PostScript language for printing systems [1], the Raster Content portion of the ISO Office Document Architecture and Interchange Format [12], the future CCITT color facsimile standard, and the European ETSI videotex standard [9].

Standardization Schedule

JPEG's ISO standard will be divided into two parts. Part 1 [2] will specify the four modes of operation, the different codecs specified for those modes, and the interchange format. It will also contain a substantial informational section on implementation guidelines. Part 2 [3] will specify the compliance tests which will determine whether an implementation of an encoder or decoder specified in Part 1 conforms to the standard.

There are two key balloting phases in the ISO standardization process: a Committee Draft (CD) is balloted to determine promotion to Draft International Standard (DIS), and a DIS is balloted to determine promotion to International Standard (IS). Each ballot requires four to six months. JPEG's Part 1 began CD ballot in February 1991, and Part 2 is expected to begin CD ballot by June 1991.

Though there is no guarantee that the first ballot of each phase will result in promotion to the next, JPEG's CD Part 1 contains no technical changes (other than some minor corrections) from JPEG's latest Technical Specification [13]. Successive revisions of the Technical Specification were widely distributed and subjected to informal review in many forums throughout 1990, and yet the technical content has been stable for nearly a year.

Conclusions

The emerging JPEG continuous-tone image compression standard is not a panacea that will solve the myriad issues which must be addressed before digital images will be fully integrated within all the applications that will ultimately benefit from them. For example, if two applications cannot exchange uncompressed images because they use incompatible color spaces, aspect ratios, dimensions, etc., then a common compression method will not help.

However, a great many applications are "stuck" because of storage or transmission costs, because of argument over which (nonstandard) compression method to use, or because VLSI codecs are too expensive due to low volumes. For these applications, the thorough technical evaluation, testing, selection, validation, and documentation work which JPEG committee members have performed is expected to soon yield an approved international standard that will withstand the tests of quality and time. As diverse imaging applications become increasingly implemented on open, networked computing systems, the ultimate measure of the committee's success will be when JPEG-compressed digital images come to be regarded and even taken for granted as "just another data type," as text and graphics are today.

For more information

Regarding the proposed standard itself, instructions on how to obtain the ISO Committee Draft Part 1, the JPEG Technical Specification, which preceded it, and other key documents as they become available can be obtained by writing the author at the following address:

Digital Equipment Corporation
146 Main Street, MLO5-2/G1
Maynard, MA 01754-2571

Floppy disks containing uncompressed, compressed, and reconstructed data for the purpose of informally validating whether an encoder or decoder implementation conforms to the proposed standard are available. Thanks to the following JPEG committee member and his company who have agreed to provide these for a nominal fee on behalf of the committee until arrangements can be made for ISO

FIGURE 11.

Progressive Build-up, showing Spectral Selection vs. Successive Approximation
a) Original Image (CCIR-601 Format: YUV, 720 × 576 Y samples)
b) Spectral Selection
b1. DC coefficients only: 0.19 bits/pixel
b2. Addition of 1 AC coefficient: 0.32 bits/pixel
b3. Addition of 2d. AC coefficient: 0.43 bits/pixel
b4. Addition of 3d.-9th AC coefficients: 0.96 bits/pixel
c) Successive Approximation
c1. 7 MSBs of DC coefficient: 0.15 bits/pixel
c2. Addition of 5 MSBs of AC coefficients: 0.3 bits/pixel
c3. Addition of 6th MSB of AC coefficients: 0.49 bits/pixel
c4. Addition of 7th MSB of AC coefficients: 0.8 bits/pixel

to provide them:

Eric Hamilton
C-Cube Microsystems
399-A W. Trimble Road
San Jose, CA 95131

Acknowledgments
The following longtime JPEG core members have spent untold hours (usually in addition to their "real jobs") to make this collaborative international effort succeed. Each has made specific substantive contributions to the JPEG proposal: Aharon Gill (Zoran, Israel), Eric Hamilton (C-Cube, USA), Alain Leger (CCETT, France), Adriaan Ligtenberg (Storm, USA), Herbert Lohscheller (ANT, Germany), Joan Mitchell (IBM, USA), Michael Nier (Kodak, USA), Takao Omachi (NEC, Japan), William Pennebaker (IBM, USA), Henning Poulsen (KTAS, Denmark), and Jorgen Vaaben (AutoGraph, Denmark). The leadership efforts of Hiroshi Yasuda (NTT, Japan), the Convenor of JTC1/SC2/WG8 from which JPEG was spawned, Istvan Sebestyen (Siemens, Germany), the Special Rapporteur from CCITT SGVIII, and Graham Hudson (British Telecom, U.K.), former JPEG chair and founder of the effort which became JPEG. The author regrets that space does not permit recognition of the many other individuals who contributed to JPEG's work.

William Pennebaker and colleagues at IBM Research provided the processed JPEG test images in Figure 11.

The author's role within JPEG has been supported in a great number of ways by Digital Equipment Corporation.

References
1. Adobe Systems Inc. *PostScript Language Reference Manual.* Second Ed. Addison Wesley, Menlo Park, Calif. 1990.
2. Digital Compression and Coding of Continuous-Tone Still Images, Part 1, Requirements and Guidelines. ISO/IEC JTC1 Committee Draft 10918-1, Feb. 1991.
3. Digital Compression and Coding of Continuous-Tone Still Images, Part 2, Compliance Testing. ISO/IEC JTC1 Committee Draft 10918-2. To be published Summer 1991.
4. Encoding parameters of digital television for studios. CCIR Recommendations, Recommendation 601, 1982.
5. Hudson, G.P. The development of photographic videotex in the UK. In *Proceedings of the IEEE Global Telecommunications Conference, IEEE Communications Society,* 1984, pp. 319–322.
6. Hudson, G.P., Yasuda, H., and Sebestyen, I. The international standardization of a still picture compression technique. In *Proceedings of the IEEE Global Telecommunications Conference, IEEE Communications Society* (Nov. 1988), pp. 1016–1021.
7. Huffman, D.A. A method for the construction of minimum redundancy codes. In *Proceedings IRE,* vol. 40, 1962, pp. 1098–1101.
8. Leger, A. Implementations of fast discrete cosine transform for full color videotex services and terminals. In *Proceedings of the IEEE Global Telecommunications Conference, IEEE Communications Society* (1984), pp. 333–337.
9. Leger, A., Omachi, T., and Wallace, G. The JPEG still picture compression algorithm and its applications. *Optical Eng.* To be published.
10. Leger, A., Mitchell, M., and Yamazaki, Y. Still picture compression algorithms evaluated for international standardization. In *Proceedings of the IEEE Global Telecommunications Conference, IEEE Communications Society* (Nov. 1988), pp. 1028–1032.
11. Lohscheller, H. A subjectively adapted image communication system. *IEEE Trans. Commun. COM-32* (Dec. 1984), 1316–1322.
12. Office Document Architecture (ODA) and Interchange Format, Part 7: Raster Graphics Content Architectures. ISO/IEC JTC1 International Standard 8613-7.
13. Pennebaker, W.B., JPEG Tech. Specification, Revision 8, Informal working paper JPEG-8-R8, Aug. 1990.
14. Pennebaker, W.B., Mitchell, J.L., et. al. Arithmetic coding articles. *IBM J. Res. Dev. 32,* 6 Special Issue (Nov. 1988), 717–774.
15. Rao, K.R., and Yip, P. *Discrete Cosine Transform—Algorithms, Advantages, Applications.* Academic Press, Inc. London, 1990.
16. Standardization of Group 3 facsimile apparatus for document transmission. CCITT Recommendations, Fascicle VII.2, Recommendation T.4, 1980.
17. Wallace, G.K. Overview of the JPEG (ISO/CCITT) still image compression standard. Image Processing Algorithms and Techniques. In *Proceedings of the SPIE,* vol. 1244 (Feb. 1990), pp. 220–233.
18. Wallace, G., Vivian, R., and Poulsen, H. Subjective testing results for still picture compression algorithms for international standardization. In *Proceedings of the IEEE Global Telecommunications Conference, IEEE Communications Society* (Nov. 1988), 1022–1027.

CR Categories and Subject Descriptors: I.4.2 [**Image Processing**]: Compression (coding)—*Approximate methods, exact coding*

General Terms: Design, Standardization

Additional Key Words and Phrases: JPEG

About the Author:
GREGORY K. WALLACE is Manager of Workstations Multimedia Advanced Product Development and is Convenor of the ISO/IEC JTC1/SC2/WG10 (JPEG) committee at Digital Equipment Corporation. His research interests include compression, signal processing, and integration of real-time video and audio into workstations. **Author's Present Address:** Digital Equipment Corporation, 146 Main Street, ML05-2/G1, Maynard MA 01754-2571. email: wallace@gauss.enet.dec.com

NeXTstep: Putting JPEG to Multiple Uses

Greg Cockroft and Leo Hourvitz

NeXTstep, the standard operating environment on NeXT computers, is designed to support a wide spectrum of application development. NeXT has included support for the Baseline System portion of the JPEG draft in the 2.0 release of NeXTstep.

Software Still-Frame Image Compression

The standard image format within NeXTstep is TIFF (Tag Image File Format). NeXTstep supports the use of TIFF files through the NXImage class. This is a class of an object-oriented C language called Objective C, upon which the NeXTstep Application Toolkit is based. In the 2.0 release of NeXTstep, we have added JPEG support to all of our TIFF reading and writing facilities. JPEG-compressed TIFF files are read transparently to the application; thus, all applications that use the NXImage class (such as MediaView) can now read JPEG-compressed TIFF files. This implementation is done in software and runs on the main processor of all NeXT computers; no additional hardware support is needed. The decompression and imaging of a 24-bit 640 × 480 image takes less than ten seconds on a NeXT computer with a Motorola 68040 main processor; further optimization and the usual advances in processors will continue to shrink the required time. The availability of this software implementation on all NeXT computers means that users can safely exchange JPEG-compressed files with each other. NeXTstep uses Display PostScript to display the decompressed images on the screen—allowing all NeXT computers to display 24-bit color images whether they have 2-bit grayscale, 12-bit color, or 24-bit color displays. To be compatible with TIFF file readers on other machines, NeXTstep follows the TIFF extensions proposed by C-Cube Microsystems. Thus, TIFF files can be transferred to other JPEG-equipped systems which also follow these extensions.

NeXTdimension: Full-Motion Video Compression

On the recently introduced NeXTdimension graphics board, NeXT has included hardware JPEG processing which allows standard-resolution video to be compressed or decompressed at real-time rates. (NeXTstep 2.1 is currently shipping: this includes all video NeXTdimension capabilities with the exception of JPEG hardware support. JPEG hardware support will not be available until the NeXTstep 2.2 release in the fourth quarter of 1991.)

In combination with the video input and DMA channels provided on the NeXTdimension board, 640 × 480 video frames can be captured from any standard video source, compressed, and transferred to hard disks at a rate of 30 frames/second. (NeXTdimension can also display the video frames in a subwindow of its megapixel display at the same time.) Playback from disk works in a similar manner: A previously recorded video sequence can be read from disk, decompressed and displayed in a subwindow of the megapixel display as well as directed to a standard video output jack. The data rate for video compressed with JPEG varies. It is dependent on the source material and the quantization tables. "Nice looking" video is within the range of the SCSI disk write speed of NeXT computers—approximately 600 kilobytes per second—with standard disks. ("Nice looking" means it is apparent the video is compressed, but the artifacts are not objectionable.) "High quality" video will require 1 megabyte per second, requiring fast SCSI disks or caching of short video sequences in large memory buffers. Another alternative is to lower the frame rate, allowing higher quality frames while keeping the data rate within the range of standard SCSI disks.

This digital video capability will allow for the implementation of applications that contain video sequences as one of their many data items. Video need not be fetched from a special-purpose external device; it can be part of a disk-based document like any other data type. Direct access to any video sequence will be perfect for interactive video applications.

Future Work

These capabilities illustrate how computers will be able to handle video information using JPEG and other image compression algorithms. For instance, NeXT is currently experimenting with using JPEG compression on low-resolution frames to get a very low data-rate video system suitable for use in video mail and other applications where full-resolution video is not required. JPEG and the related MPEG draft standards described in a later article (see "MPEG: A Video Compression Standard for Multimedia Applications," in this issue) are open and flexible enough that they will continue to inspire clever systems builders to find more ways to put them to work. *(continued on page 116)*

Wavelets and Signal Processing

OLIVIER RIOUL and MARTIN VETTERLI

Wavelet theory provides a unified framework for a number of techniques which had been developed independently for various signal processing applications. For example, multiresolution signal processing, used in computer vision; subband coding, developed for speech and image compression; and wavelet series expansions, developed in applied mathematics, have been recently recognized as different views of a single theory.

In fact, wavelet theory covers quite a large area. It treats both the continuous and the discrete-time cases. It provides very general techniques that can be applied to many tasks in signal processing, and therefore has numerous potential applications.

In particular, the Wavelet Transform (WT) is of interest for the analysis of *non-stationary* signals, because it provides an alternative to the classical Short-Time Fourier Transform (STFT) or Gabor transform [GAB46, ALL77, POR80]. The basic difference is as follows. In contrast to the STFT, which uses a single analysis window, the WT uses short windows at high frequencies and long windows at low frequencies. This is in the spirit of so-called "constant-Q" or constant relative bandwidth frequency analysis. The WT is also related to time-frequency analysis based on the Wigner-Ville distribution [FLA89, FLA90, RIO90a].

For some applications it is desirable to see the WT as a signal decomposition onto a set of basis functions. In fact, basis functions called *wavelets* always underlie the wavelet analysis. They are obtained from a single prototype wavelet by dilations and contractions (scal-

1053-5888/91/1000-0014$1.00©IEEE

ings) as well as shifts. The prototype wavelet can be thought of as a bandpass filter, and the constant-Q property of the other bandpass filters (wavelets) follows because they are scaled versions of the prototype.

Therefore, in a WT, the notion of *scale* is introduced as an alternative to frequency, leading to a so-called *time-scale representation.* This means that a signal is mapped into a time-scale plane (the equivalent of the time-frequency plane used in the STFT).

There are several types of wavelet transforms, and, depending on the application, one may be preferred to the others. For a continuous input signal, the time and scale parameters can be continuous [GRO89], leading to the Continuous Wavelet Transform (CWT). They may as well be discrete [DAU88, MAL89b, MEY89, DAU90a], leading to a Wavelet Series expansion. Finally, the wavelet transform can be defined for discrete-time signals [DAU88, RIO90b, VET90b], leading to a Discrete Wavelet Transform (DWT). In the latter case it uses multirate signal processing techniques [CRO83] and is related to subband coding schemes used in speech and image compression. Notice the analogy with the (Continuous) Fourier Transform, Fourier Series, and the Discrete Fourier Transform.

Wavelet theory has been developed as a unifying framework only recently, although similar ideas and constructions took place as early as the beginning of the century [HAA10, FRA28, LIT37, CAL64]. The idea of looking at a signal at various scales and analyzing it with various resolutions has in fact emerged independently in many different fields of mathematics, physics and engineering. In the mid-eighties, researchers of the "French school," lead by a geophysicist, a theoretical physicist and a mathematician (namely, Morlet, Grossmann, and Meyer), built strong mathematical foundations around the subject and named their work "*Ondelettes*" (Wavelets). They also interacted considerably with other fields.

The attention of the signal processing community was soon caught when Daubechies and Mallat, in addition to their contribution to the theory of wavelets, established connections to discrete signal processing results [DAU88], [MAL89a]. Since then, a number of theoretical, as well as practical contributions have been made on various aspects of WT's, and the subject is growing rapidly [WAV89], [IT92].

The present paper is meant both as a review and as a tutorial. It covers the main definitions and properties of wavelet transforms, shows connections among the various fields where results have been developed, and focuses on signal processing applications. Its purpose is to present a simple, synthetic view of wavelet theory, with an easy-to-read, non-rigorous flavor. An extensive bibliography is provided for the reader who wants to go into more detail on a particular subject.

NON-STATIONARY SIGNAL ANALYSIS

The aim of signal analysis is to extract relevant information from a signal by transforming it. Some methods make *a priori* assumptions on the signal to be analyzed; this may yield sharp results if these assumptions are valid, but is obviously not of general applicability. In this paper we focus on methods that are applicable to any general signal. In addition, we consider invertible transformations. The analysis thus unambiguously represents the signal, and more involved operations such as parameter estimation, coding and pattern recognition can be performed on the "transform side," where relevant properties may be more evident.

Such transforms have been applied to *stationary* signals, that is, signals whose properties do not evolve in time (the notion of stationarity is formalized precisely in the statistical signal processing literature). For such signals $x(t)$, the natural "stationary transform" is the well-known Fourier transform [FOU88]:

$$X(f) = \int_{-\infty}^{+\infty} x(t)\, e^{-2j\pi f t} dt \qquad (1)$$

The analysis coefficients $X(f)$ define the notion of global frequency f in a signal. As shown in (1), they are computed as inner products of the signal with sinewave basis functions of infinite duration. As a result, Fourier analysis works well if $x(t)$ is composed of a few stationary components (e.g., sinewaves). However, any abrupt change in time in a non-stationary signal $x(t)$ is spread out over the whole frequency axis in $X(f)$. Therefore, an analysis adapted to *nonstationary* signals requires more than the Fourier Transform.

The usual approach is to introduce time dependency in the Fourier analysis while preserving linearity. The idea is to introduce a "local frequency" parameter (local in time) so that the "local" Fourier Transform looks at the signal through a window over which the signal is approximately stationary. Another, equivalent way is to modify the sinewave basis functions used in the Fourier Transform to basis functions which are more concentrated in time (but less concentrated in frequency).

SCALE VERSUS FREQUENCY

The Short-Time Fourier Transform: Analysis with Fixed Resolution.

The "instantaneous frequency" [FLA89] has often been considered as a way to introduce frequency de-

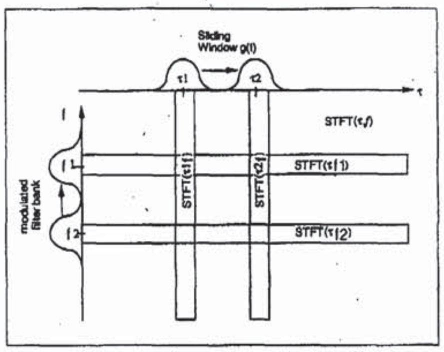


Fig. 1. Time-frequency plane corresponding to the Short-Time Fourier Transform. It can be seen either as a succession of Fourier Transforms of a windowed segment of the signal (vertical stripes) or as a modulated analysis filter bank (horizontal stripes).

pendence on time. If the signal is not narrow-band, however, the instantaneous frequency averages different spectral components in time. To become accurate in time, we therefore need a *two-dimensional* time-frequency representation $S(t,f)$ of the signal $x(t)$ composed of spectral characteristics depending on time, the local frequency f being defined through an appropriate definition of $S(t,f)$. Such a representation is similar to the notation used in a musical score, which also shows "frequencies" played in time.

The Fourier Transform (1) was first adapted by Gabor [GAB46] to define $S(t,f)$ as follows. Consider a signal $x(t)$, and assume it is stationary when seen through a window $g(t)$ of limited extent, centered at time location τ. The Fourier Transform (1) of the windowed signals $x(t)\,g^*(t-\tau)$ yields the Short-Time Fourier Transform (STFT)

$$\mathrm{STFT}(\tau, f) = \int x(t)\, g^*(t-\tau)\, e^{-2j\pi f t}\, dt \qquad (2)$$

which maps the signal into a two-dimensional function in a time-frequency plane (τ,f). Gabor originally only defined a synthesis formula, but the analysis given in (2) follows easily.

The parameter f in (2) is similar to the Fourier frequency and many properties of the Fourier transform carry over to the STFT. However, the analysis here depends critically on the choice of the window $g(t)$.

Figure 1 shows vertical stripes in the time-frequency plane, illustrating this "windowing of the signal" view of the STFT. Given a version of the signal windowed around time t, one computes all "frequencies" of the STFT.

An alternative view is based on a filter bank interpretation of the same process. At a given frequency f, (2) amounts to filtering the signal "at all times" with a bandpass filter having as impulse response the window function modulated to that frequency. This is shown as the horizontal stripes in Fig. 1. Thus, the STFT may be seen as a modulated filter bank [ALL77], [POR80].

From this dual interpretation, a possible drawback related to the time and frequency resolution can be shown. Consider the ability of the STFT to discriminate between two pure sinusoids. Given a window function $g(t)$ and its Fourier transform $G(f)$, define the "bandwidth" Δf of the filter as

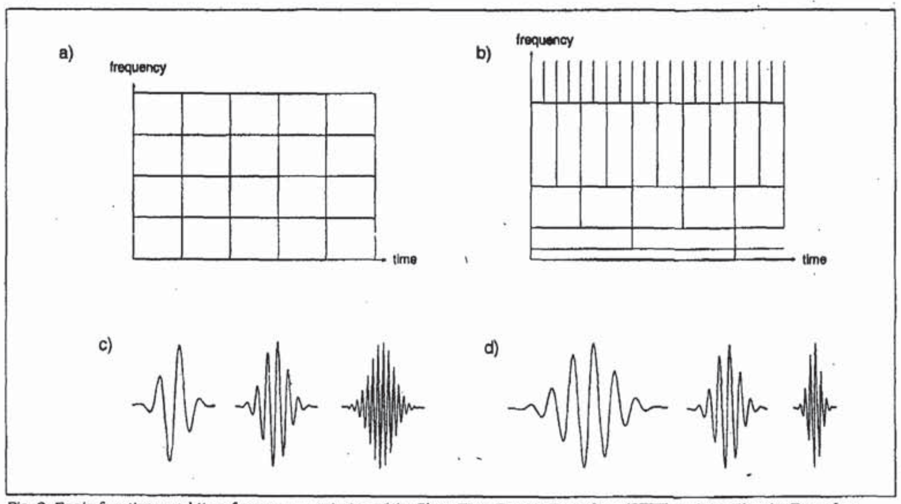


Fig. 2. Basis functions and time-frequency resolution of the Short-Time Fourier Transform (STFT) and the Wavelet Transform (WT). The tiles represent the essential concentration in the time-frequency plane of a given basis function. (a) Coverage of the time-frequency plane for the STFT, (b) for the WT. (c) Corresponding basis functions for the STFT, (d) for the WT ("wavelets").

$$\Delta f^2 = \frac{\int f^2 \, |G(f)|^2 \, df}{\int |G(f)|^2 \, df} \quad (3)$$

where the denominator is the energy of $g(t)$. Two sinusoids will be discriminated only if they are more than Δf apart (This is an rms measure, and others are possible). Thus, the resolution in frequency of the STFT analysis is given by Δf. Similarly, the spread in time is given by Δt as

$$\Delta t^2 = \frac{\int t^2 |g(t)|^2 \, dt}{\int |g(t)|^2 \, dt} \quad (4)$$

where the denominator is again the energy of $g(t)$. Two pulses in time can be discriminated only if they are more than Δt apart.

Now, resolution in time and frequency cannot be arbitrarily small, because their product is lower bounded.

$$\text{Time - Bandwidth product} = \Delta t \, \Delta f \geq \frac{1}{4\pi} \quad (5)$$

This is referred to as the uncertainty principle, or Heisenberg inequality. It means that one can only trade time resolution for frequency resolution, or *vice versa*. Gaussian windows are therefore often used since they meet the bound with equality [GAB46].

More important is that once a window has been chosen for the STFT, then the time-frequency resolution given by (3), (4) is *fixed* over the entire time-frequency plane (since the same window is used at all frequencies). This is shown in Fig. 2a, while Fig. 2c shows the associated basis functions of the STFT. For example, if the signal is composed of small bursts associated with long quasi-stationary components, then each type of component can be analyzed with good time resolution or frequency resolution, but not both.

The Continuous Wavelet Transform: A Multiresolution Analysis.

To overcome the resolution limitation of the STFT, one can imagine letting the resolution Δt and Δf vary in the time-frequency plane in order to obtain a multiresolution analysis. Intuitively, when the analysis is viewed as a filter bank, the time resolution must increase with the central frequency of the analysis filters. We therefore impose that Δf is proportional to f, or

$$\frac{\Delta f}{f} = c \quad (6)$$

where c is a constant. The analysis filter bank is then composed of band-pass filters with constant relative bandwidth (so-called "constant-Q" analysis). Another way to say this is that instead of the frequency responses of the analysis filter being regularly spaced over the frequency axis (as for the STFT case), they are regularly spread in a logarithmic scale (see Fig. 3). This kind of filter bank is used, for example, for modeling the frequency response of the cochlea situated in the inner ear and is therefore adapted to auditory perception, e.g. of music: filters satisfying (6) are naturally distributed into octaves.

When (6) is satisfied, we see that Δf and therefore also Δt changes with the center frequency of the analysis filter. Of course, they still satisfy the Heisenberg inequality (5), but now, the time resolution becomes arbitrarily good at high frequencies, while the frequency resolution becomes arbitrarily good at low frequencies. For example, two very close short bursts can always be eventually separated in the analysis by going up to

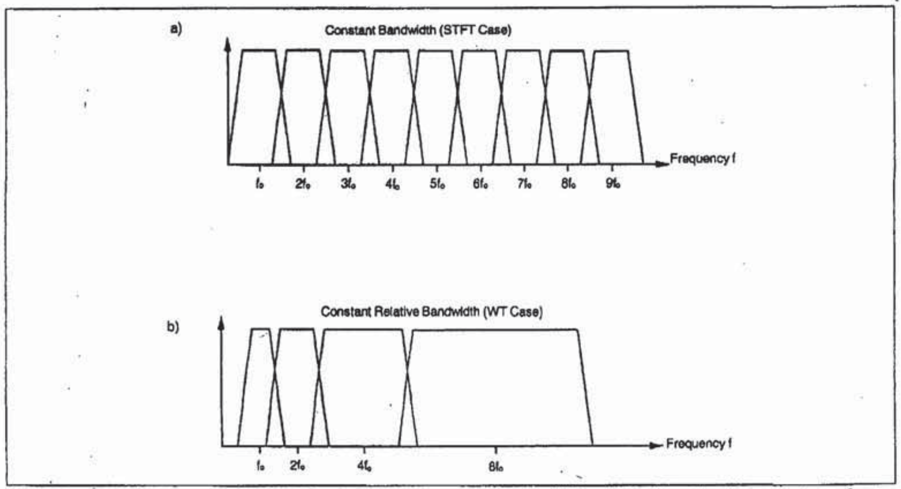

a)
Constant Bandwidth (STFT Case)
Frequency f
f_0 $2f_0$ $3f_0$ $4f_0$ $5f_0$ $6f_0$ $7f_0$ $8f_0$ $9f_0$
b)
Constant Relative Bandwidth (WT Case)
Frequency f
f_0 $2f_0$ $4f_0$ $8f_0$

Fig. 3. Division of the frequency domain (a) for the STFT (uniform coverage) and (b) for the WT (logarithmic coverage).

Box 1: The Notion of Scale and Resolution

First, recall that when a function $f(t)$ is scaled:

$f(t) \rightarrow f(at)$, where $a > 0$,

then it is contracted if $a > 1$ and expanded if $a < 1$ Now, the CWT can be written either as

$$CWT_x(\tau,a) = \frac{1}{\sqrt{a}} \int x(t)\, h^*(\frac{t-\tau}{a})dt \qquad (B1.1)$$

or, by a change of variable, as

$$CWT_x(\tau,a) = \sqrt{a} \int x(at)\, h^*(t - \frac{\tau}{a})dt \qquad (B1.2)$$

The interpretation of (B1.1) is that as the scale increases, the filter impulse response $h\left(\frac{t-\tau}{a}\right)$ becomes spread out in time, and takes only long-time behavior into account. Equivalently, (B1.2) indicates that as the scale grows, an increasingly contracted version of the signal is seen through a constant length filter. That is, the scale factor a has the interpretation of the scale in maps. Very large scales mean global views, while very small scales mean detailed views.

A related but different notion is that of resolution. The resolution of a signal is linked to its frequency content. For example, lowpass filtering a signal keeps its scale, but reduces its resolution.

Scale change of continuous time signals does not alter their resolution, since the scale change can be reversed. However, in discrete-time signals, increasing the scale in the analysis involves subsampling, which automatically reduces the resolution. Decreasing the scale (which involves upsampling) can be undone, and does not change the resolution. The interplay of scale and resolution changes in discrete-time signals is illustrated in Fig. 9 and fully explained in [RIO90b], [VET90b].

higher analysis frequencies in order to increase time resolution (see Fig. 2b). This kind of analysis of course works best if the signal is composed of high frequency components of short duration plus low frequency components of long duration, which is often the case with signals encountered in practice.

A generalization of the concept of changing resolution at different frequencies is obtained with so-called "wavelet packets" [WIC89], where arbitrary time-frequency resolutions (within the uncertainty bound (5)) are chosen depending on the signal.

The *Continuous Wavelet Transform* (CWT) exactly follows the above ideas while adding a simplification: all impulse responses of the filter bank are defined as *scaled* (i.e. stretched or compressed) versions of the same prototype $h(t)$, i.e.,

$$h_a(t) = \frac{1}{\sqrt{|a|}}\, h(\frac{t}{a})$$

where a is a *scale factor* (the constant $1/\sqrt{|a|}$ is used for energy normalization). This results in the definition of the CWT:

$$CWT_x(\tau, a) = \frac{1}{\sqrt{|a|}} \int x(t)\, h^*\left(\frac{t-\tau}{a}\right)dt \qquad (7)$$

Since the same prototype $h(t)$, called the *basic wavelet*, is used for all of the filter impulse responses, no specific scale is privileged, i.e. the wavelet analysis is self-similar at all scales. Moreover, this simplification is useful when deriving mathematical properties of the CWT.

To make the connection with the modulated window used in the STFT clearer, the basic wavelet $h(t)$ in (7) could be chosen as a modulated window [GOU84, GRO84, GRO89]

$$h(t) = g(t)\, e^{-2j\pi f_0 t}$$

Then the frequency responses of the analysis filters indeed satisfy (6) with the identification

$$a = \frac{f_0}{f}$$

But more generally, $h(t)$ can be any band-pass function and the scheme still works. In particular one can dispense with complex-valued transforms and deal only with real-valued ones.

It is important to note that here, the local frequency $f = a f_0$ has little to do with that described for the STFT: indeed, it is associated with the scaling scheme (see Box 1). As a result, this local frequency, whose definition depends on the basic wavelet, is no longer linked to frequency modulation (as was the case for the STFT) but is now related to time-scalings. This is the reason why the terminology "scale" is often preferred to "frequency" for the CWT, the word "frequency" being reserved for the STFT. Note that we define *scale* in wavelet analysis like the scale in geographical maps: since the filter bank impulse responses in (7) are dilated as scale increases, large scale corresponds to contracted signals, while small scale corresponds to dilated signals.

WAVELET ANALYSIS AND SYNTHESIS

Another way to introduce the CWT is to define *wavelets* as basis functions. In fact, basis functions already appear in the preceding definition (7) when one sees it as an inner product of the form

$$CWT_x(\tau, a) = \int x(t)\, h^*_{a,\tau}(t)\, dt$$

which measures the "similarity" between the signal and the basis functions

Box 2: STFTs and CWTs as Cross-Ambiguity Functions

The inner product is often used as a similarity measurement, and because both STFT's and CWT's are inner products, they appear in several detection/estimation problems. Consider, for example, the problem of estimating the location and velocity of some target in radar or sonar applications. The estimation procedure consists in first emitting a known signal $h(t)$. In the presence of a target, this signal will return to the source (received signal $x(t)$) with a certain delay τ, due to the target's location, and a certain distortion (Doppler effect), due to the target's velocity.

For narrow-band signals, the Doppler effect amounts to a single frequency shift f_0 and the characteristics of the target will be determined by maximizing the cross-correlation function (called "narrow-band cross-ambiguity function") [WOO53]

$$\int x(t)\, h(t-\tau)\, e^{-2j\pi f_0 t}\, dt = \mathrm{STFT}(\tau, f)$$

For wide-band signals, however, the Doppler frequency shift varies in the signal's spectrum, causing a stretching or a compression in the signal. The estimator thus becomes the "wide-band cross-ambiguity function" [SPE67], [AUS90]

$$\frac{1}{\sqrt{|a|}} \int x(t)\, h\!\left(\frac{t-\tau}{a}\right) dt = \mathrm{CWT}_x(\tau, a)$$

As a result, in both cases, the "maximum likelihood" estimator takes the form of a STFT or a CWT, i.e. of an inner product between the received signal and either STFT or wavelet basis functions. The basis function which best fits the signal is used to estimate the parameters.

Note that, although the wide-band cross-ambiguity function is a CWT, for physical reasons, the dilation parameter a stays on the order of magnitude of 1, whereas it may cover several octaves when used in signal analysis [FLA89].

$$h_{a,\tau} = \frac{1}{\sqrt{a}}\, h\!\left(\frac{t-\tau}{a}\right)$$

called *wavelets*. The wavelets are scaled and translated versions of the basic wavelet prototype $h(t)$ (see Fig. 2d).

Of course, basis functions can be considered for the STFT as well. For both the STFT and the CWT, the sinewaves basis functions of the Fourier Transform are replaced by more localized reference signals labelled by time and frequency (or scale) parameters. In fact both transforms may be interpreted as special cases of the cross-ambiguity function used in radar or sonar processing (see Box 2).

The wavelet analysis results in a set of wavelet coefficients which indicate how close the signal is to a particular basis function. Thus, we expect that any general signal can be represented as a decomposition into wavelets, i.e. that the original waveform is synthesized by adding elementary building blocks, of constant shape but different size and amplitude. Another way to say this is that we want the continuously labelled wavelets $h_{a,\tau}(t)$ to behave just like an *orthogonal basis* [MEY90]. The analysis is done by computing inner products, and the synthesis consists of summing up all the orthogonal projections of the signal onto the wavelets.

$$x(t) = c \iint_{a>0} \mathrm{CWT}(\tau, a)\, h_{a,\tau}(t)\, \frac{da\,d\tau}{a^2} \qquad (8)$$

where c is a constant that depends only on $h(t)$. The measure in this integration is formally equivalent to $dt\,df$ [GOU84]. We have assumed here that both signal and wavelets are either real-valued or complex analytic so that only positive dilations $a > 0$ have to be taken into account. Otherwise (8) is more complicated [GRO84].

Of course, the $h_{a,\tau}(t)$ are certainly not orthogonal since they are very redundant (they are defined for continuously varying a and τ). But surprisingly, the reconstruction formula (8) is indeed satisfied whenever $h(t)$ is of finite energy and *band pass* (which implies that it oscillates in time like a short wave, hence the name "wavelet"). More precisely, if $h(t)$ is assumed sufficiently regular, then the reconstruction condition is $\int h(t)\, dt = 0$.

Note that the reconstruction takes place only in the sense of the signal's energy. For example, a signal may be reconstructed only with zero mean since $\int h(t)\, dt = 0$. In fact the type of convergence of (8) may be strengthened and is related to the numerical robustness of the reconstruction [DAU90a].

Similar reconstruction can be considered for the STFT, and the similarity is remarkable [DAU90a]. However, in the STFT case, the reconstruction condition is less restrictive: only finite energy of the window is required.

SCALOGRAMS

The *spectrogram*, defined as the square modulus of the STFT, is a very common tool in signal analysis because it provides a distribution of the energy of the signal in the time-frequency plane. A similar distribution can be defined in the wavelet case. Since the CWT behaves like an orthonormal basis decomposition, it can be shown that it is isometric [GRO84], i.e., it preserves energy. We have

$$\iint |\mathrm{CWT}(\tau, a)|^2 \frac{d\tau\, da}{a^2} = E_x$$

where $E_x = \int |x(t)|^2\, dt$ is the energy of the signal $x(t)$. This leads us to define the *wavelet spectrogram*, or *scalogram*, as the squared modulus of the CWT. It is a distribution of the energy of the signal in the time-scale

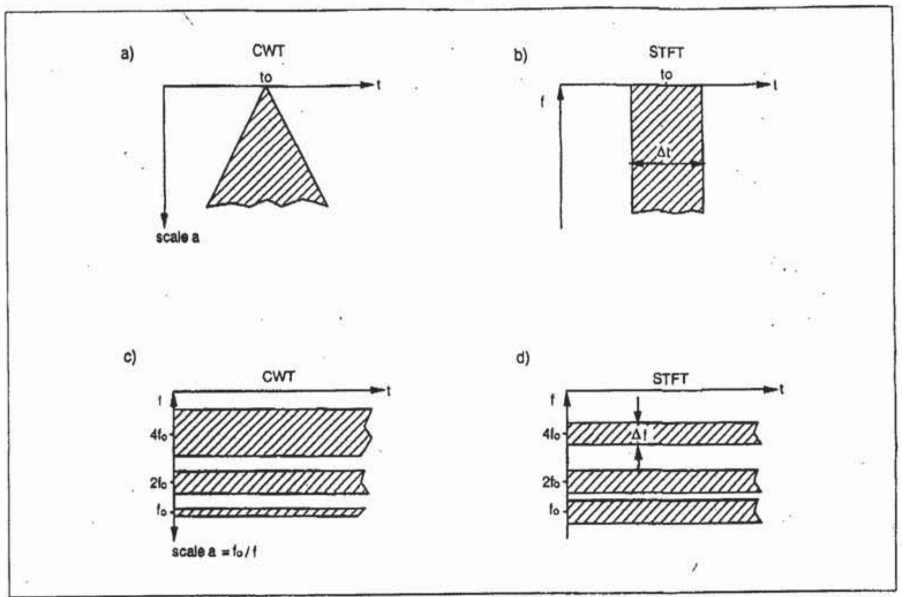


Fig. 4. Regions of influence of a Dirac pulse at $t=t_0$ (a) for the CWT and (b) for the STFT; as well as of three sinusoids (of frequencies f_0, $2f_0$, $4f_0$) for (c) the CWT and (d) the STFT.

plane, associated with measure $\frac{d\tau . da}{a^2}$, and thus expressed in power per frequency unit, like the spectrogram. However, in contrast to the spectrogram, the energy of the signal is here distributed with different resolutions according to Fig. 2b.

Figure 4 illustrates differences between a scalogram and a spectrogram. Figure 4a shows that the influence of the signal's behavior around $t = t_0$ in the analysis is limited to a cone in the time-scale plane; it is therefore very "localized" around t_0 for small scales. In the STFT case, the corresponding region of influence is as large as the extent of the analysis window over all frequencies, as shown in Fig. 4b. Moreover, since the time-scale analysis is logarithmic in frequency, the area of influence of some pure frequency f_0 in the signal increases with f_0 in a scalogram (Fig. 4c), whereas it remains constant in a spectrogram (Fig. 4d).

Both the spectrogram and the scalogram produce a more or less easily interpretable visual two-dimensional representation of signals [GRO89], where each pattern in the time-frequency or time-scale plane contributes to the global energy of the signal. However, such an energy representation has some disadvantages, too. For example, the spectrogram, as well as the scalogram, cannot be inverted in general. Phase information is necessary to reconstruct the signal. Also, since both the spectrogram and the scalogram are bilinear functions of the analyzed signal, cross-terms appear as interferences between patterns in the time-frequency or time-scale plane [KAD91] and this may be undesirable.

In the wavelet case, it has been also shown [GRO89] that the *phase* representation more accurately reveals isolated, local bursts in a signal than the scalogram does (see Box 3).

To illustrate the above points, Fig. 5 shows some examples of spectrograms and scalograms for synthetic signals and a speech signal (see Box 3).

More involved energy representations can be developed for both time-frequency and time-scale [BER88, FLA90, RIO90a], and a link between the spectrogram, the scalogram and the Wigner-Ville distribution can be established (see Box 4).

WAVELET FRAMES AND ORTHONORMAL BASES

Discretization of Time-Scale Parameters

We have seen that the continuously labelled basis functions (wavelets) $h_{a,\tau}(t)$ behave in the wavelet analysis and synthesis just like an orthonormal basis. The following natural question arises: if we appropriately discretize the time-scale parameters a, τ, can we obtain a *true* orthonormal basis? The answer, as we shall see, is that it depends on the choice of the basic wavelet $h(t)$.

There is a natural way to discretize the time-scale parameters a, τ [DAU90a]: since two scales $a_0 < a_1$ roughly correspond to two frequencies $f_0 > f_1$, the wavelet coefficients at scale a_1 can be subsampled at $(f_0/f_1)^{th}$ the rate of the coefficients at scale a_0, according

BOX 3: Spectrograms and Scalograms

We present in Fig. 5 spectrograms and scalograms for some synthetic signals and a real signal. The signals are of length 384 samples, and the STFT uses a Gaussian-like window of length $L = 128$ samples. The scalogram is obtained with a Morlet wavelet (a complex sinusoid windowed with a Gaussian envelope) of length from 23 to 363 samples.

The horizontal axis is time in both spectrograms and scalograms. The signal is shown on the top. The vertical axis is frequency in the spectrogram (high frequencies on top) and scale in the scalogram (small scale at the top). Compare these figures with Fig. 4, which indicates the axis system used, and gives the rough behavior for Diracs and sinewaves.

First, Fig. 5.1 shows the analysis of two Diracs and two sinusoids with the STFT and the CWT. Note how the Diracs are well time-localized at high frequencies in the scalogram. Figure 5.2 shows the analysis of three starting sinusoids with different starting times (a low frequency starts first, followed by a medium and a high frequency sinewave). Figure 5.3 shows the transforms of a chirp signal. Again, the transitions are well resolved at high frequencies in the scalogram. Finally, Fig. 5.4 shows the analysis of a segment of speech signal, where the onset of voicing is seen in both representations.

Note that displaying scalograms is sometimes tricky, because parameters like display look-up tables (which map the scalogram value to a grey scale value on the screen) play an important but not always well understood role in the visual impression. Such problems are common in spectrogram displays as well.

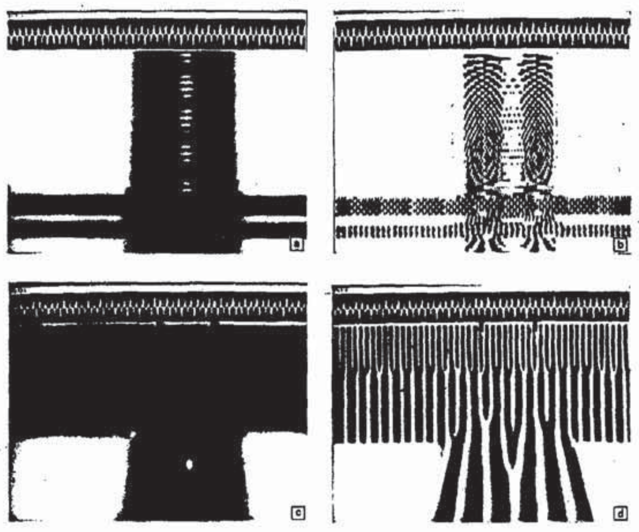


Fig. 5.1. Spectrogram and scalogram for the STFT and CWT analysis of two Dirac pulses and two sinusoids. (a) Magnitude of the STFT. (b) Phase of the STFT. (c) Amplitude of the WT. (d) Phase of the WT.

BOX 3: Spectrograms and Scalograms *(continued)*

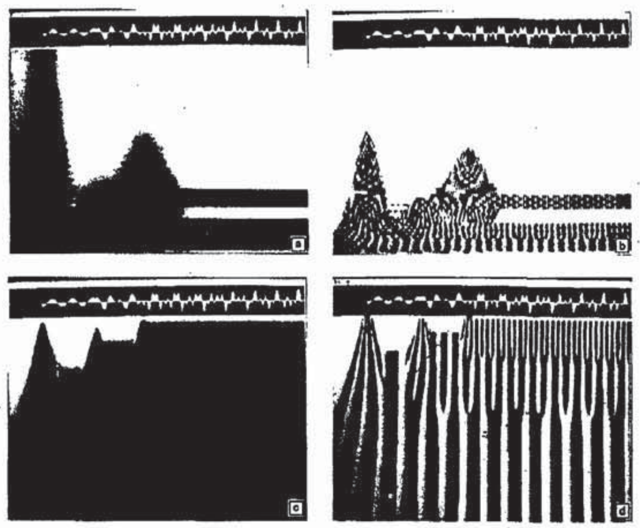


Fig. 5.2. Spectrogram and scalogram for the STFT and CWT analysis of three sinusoids with staggered starting times. The low frequency one comes first, followed by the medium and high frequency ones. (a) Magnitude of the STFT. (b) Phase of the STFT. (c) Amplitude of the WT. (d) Phase of the WT.

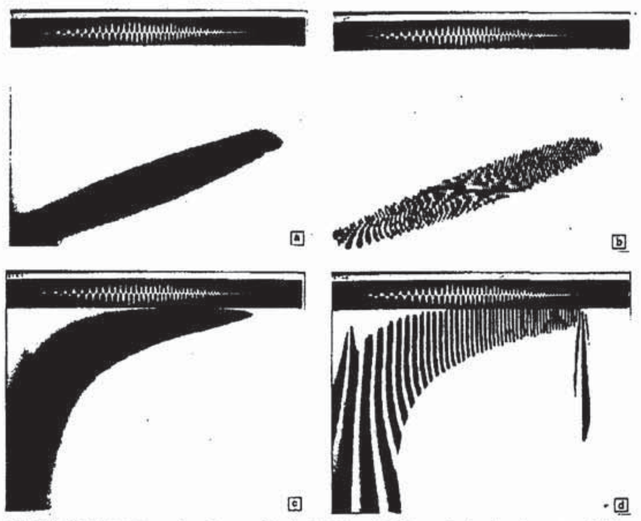


Fig. 5.3. Spectrogram and scalogram for the STFT and CWT analysis of a chirp signal. (a) Magnitude of the STFT. (b) Phase of the STFT. (c) Amplitude of the WT. (d) Phase of the WT.

BOX 3: Spectrograms and Scalograms *(continued)*

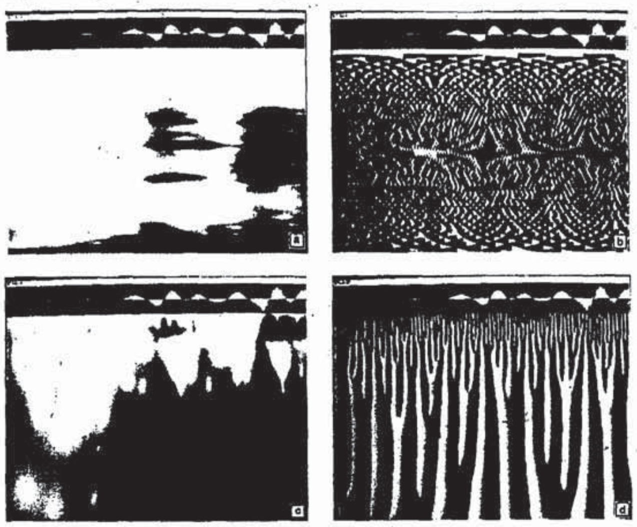


Fig. 5.4. Spectrogram and scalogram for the STFT and CWT analysis of a segment of speech, including onset of voicing. (a) Magnitude of the STFT. (b) Phase of the STFT. (c) Amplitude of the WT. (d) Phase of the WT.

to Nyquist's rule. We therefore choose to discretize the time-scale parameters on the sampling grid drawn in Fig. 7. That is, we have $a = a_0^j$ and $b = k\, a_0^j\, T$, where j and k are integers. The corresponding wavelets are

$$h_{j,k}(t) = a_0^{-j/2}\, h(a_0^{-j}\, t - kT) \qquad (9)$$

resulting in wavelet coefficients

$$c_{j,k} = \int x(t)\, h_{j,k}^{*}(t)\, dt \qquad (10)$$

An analogy is the following: assume that the wavelet analysis is like a microscope. First one chooses the magnification, that is, a_0^{-j}. Then one moves to the chosen location. Now, if one looks at very small details, then the chosen magnification is large and corresponds to j negative and large. Then, $a_0^j\, T$ corresponds to small steps, which are used to catch small details. This justifies the choice $b = k\, a_0^j\, T$ in (9).

The reconstruction problem is to find a_0, T, and $h(t)$ such that

$$x(t) = c \sum_{j} \sum_{k} c_{j,k}\, h_{j,k}(t) \qquad (11)$$

where c is a constant that does not depend on the signal (compare with (8)). Evidently, if a0 is close enough to 1 (and if T is small enough), then the wavelet functions are overcomplete. Equation (11) is then still very close to (8) and signal reconstruction takes place within non-restrictive conditions on h(t). On the other hand, if the sampling is sparse, e.g. the computation is done octave by octave ($a_0 = 2$), a true orthonormal basis will be obtained only for very special choices of h(t) [DAU90a, MEY90].

Wavelet Frames

The theory of wavelet frames [DUF52, DAU90a] provides a general framework which covers the two extreme situations just mentioned. It therefore permits one to balance (i) redundancy, i.e. sampling density in Fig. 7, and (ii) restrictions on h(t) for the reconstruction scheme (11) to work. The trade-off is the following: if the redundancy is large (high "oversampling"), then only mild restrictions are put on the basis functions (9). But if the redundancy is small (i.e., close to "critical" sampling), then the basis functions are very constrained.

The idea behind frames [DUF52] is based on the assumption that the linear operator x(t) Ã cj,k, where cj,k is defined by (10), is bounded, with bounded inverse. The family of wavelet functions is then called a frame and is such that the energy of the wavelet coefficients cj,k (sum of their square moduli) relative to that of the signal lies between two positive "frames bounds" A and B,

$$A{\cdot}E_x \le \sum_{j,k} |\, c_{j,k}\, |^2 \le B{\cdot}E_x$$

Box 4: Merging Spectrogram, Scalogram, and Wigner Distribution into a Common Class of Energy Representations

There has been considerable work in extending the spectrogram into more general time-frequency energy distributions $TF(\tau, f)$. These all have the basic property of distributing the energy of the signal all over the time-frequency plane, i.e.,

$$\iint TF(\tau, f)\, d\tau\, df = \int |x(t)|^2 dt$$

Among them, an alternative to the spectrogram for nonstationary signal analysis is the Wigner-Ville distribution [CLA80, BOU85, FLA89]

$$W_x(\tau, f) = \int x(\tau + \frac{t}{2})\, x^*(\tau - \frac{t}{2})\, e^{-2j\pi f t} dt$$

More generally, the whole class of time-frequency energy distributions has been fully described by Cohen [COH66], [COH89]: they can all be seen as smoothed (or, more precisely, correlated) versions of the Wigner-Ville distribution. The spectrogram is itself recovered when the "smoothing" function is the Wigner-Ville distribution of the analysis window!

A similar situation appears for time-scale energy distributions. For example, the scalogram can be written as [FLA90], [RIO90a]

$$|CWT(\tau, a)|^2 = \iint W_x(t, \nu)\, W_h(\frac{t-\tau}{a}, a\nu)\, dt\, d\nu$$

i.e., as some 2D "affine" correlation between the signal and the "basic" wavelet's Wigner-Ville distribution. This remarkable formula tells us that there exist strong links between Wavelet Transforms and Wigner-Ville distributions. And, as a matter of fact, it can be generalized to define the most general class of time-scale energy distributions [BER88, FLA90, RIO90a], just as in the time-frequency case.

Figure 6 shows that it is even possible to go continuously from the spectrogram of a given signal to its scalogram [FLA90, RIO90a]. More precisely, starting from the Wigner-Ville distribution, by progressively controlling Gaussian smoothing functions, one goes through a set of energy representations which either tends to the spectrogram if regular two-dimensional smoothing is used, or to the scalogram if "affine" smoothing is used. This property may allow us to decide whether or not we should choose time-scale analysis tools, rather than time-frequency ones for a given problem.

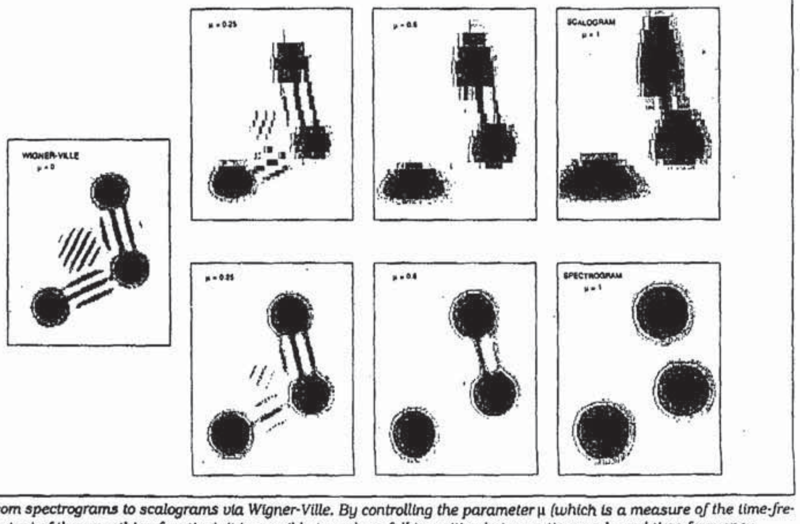


Fig. 6. From spectrograms to scalograms via Wigner-Ville. By controlling the parameter μ (which is a measure of the time-frequency extent of the smoothing function), it is possible to make a full transition between time-scale and time-frequency analyses. Here seven analyses of the same signal (composed of three Gaussian packets) are shown. Note that the best joint time-frequency resolution is attained for the Wigner-Ville distribution, while both spectrogram and scalogram (which can be thought of as smoothed versions of Wigner-Ville) provide reduced cross-term effects compared to Wigner-Ville (after [FLA90, RIO90a]).

where E_x is the energy of the signal $x(t)$.

These frame bounds can be computed from a_0, T and $h(t)$ using Daubechies' formulae [DAU90a]. Interestingly enough, they govern the accuracy of signal reconstruction by (11). More precisely, we have

$$x(t) \approx \frac{2}{A+B} \sum_j \sum_k c_{j,k}\, h_{j,k}(t)$$

with relative SNR greater than $(B/A+1)/(B/A-1)$ (see Fig. 8). The closer A and B, the more accurate the reconstruction. It may even happen that $A=B$ ("tight frame"), in which case the wavelets behave exactly like an orthonormal basis, although they may not even be linearly independent [DAU90a]! The reconstruction formula can also be made exact in the general case if one uses different synthesis functions $h'_{jk}(t)$ (which constitute the *dual frame* of the $h_{jk}(t)$s [DAU90a]).

Introduction to orthogonal wavelet bases

If a tight frame is such that all wavelets $h_{j,k}(t)$ (9) are necessary to reconstruct a general signal, then the wavelets form an *orthonormal* basis of the space of signals with finite energy [HEI90]. Recall that orthonormality means

$$\int h_{j,k}(t)\, h^*_{j',k'}(t)\, dt = \begin{cases} 1 & \text{if } j=j' \text{ and } k=k' \\ 0 & \text{otherwise} \end{cases}$$

An arbitrary signal can then be represented exactly as a weighted sum of basis functions,

$$x(t) = \sum_{j,k} c_{j,k}\, h_{j,k}(t)$$

That is, not only the basis functions $h_{j,k}(t)$ are obtained from a single prototype function $h(t)$ by means of scalings and shifts, but also they form an orthonormal basis. What is most interesting is that there do exist well-behaved functions $h(t)$ that can be used as prototype wavelets, as we shall see below. This is in sharp contrast with the STFT, where, according to the Balian-Low theorem [DAU90a], it is impossible to have orthonormal bases with functions well localized in time and frequency (that is, for which the time-bandwidth product $\Delta t\, \Delta f$ is a finite number).

Recently, the wavelet orthonormal scheme has been extended to synthesis functions $h'_{jk}(t) \neq h_{jk}(t)$, leading to so-called *biorthogonal* wavelet bases [COH90a], [VET90a], [VET90b].

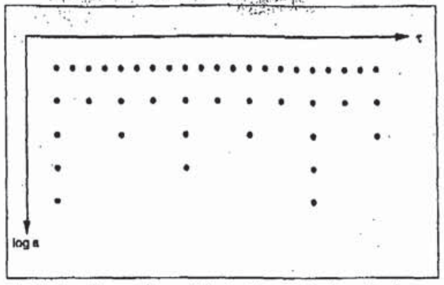


Fig. 7. Dyadic sampling grid in the time-scale plane. Each node corresponds to a wavelet basis function $h_{j,k}(t)$ with scale 2^{-j} and shift $2^{-j}k$.

THE DISCRETE TIME CASE

In this section, we first take a purely discrete-time point of view. Then, through the construction of iterated filter banks, we shall come back to the continuous-time

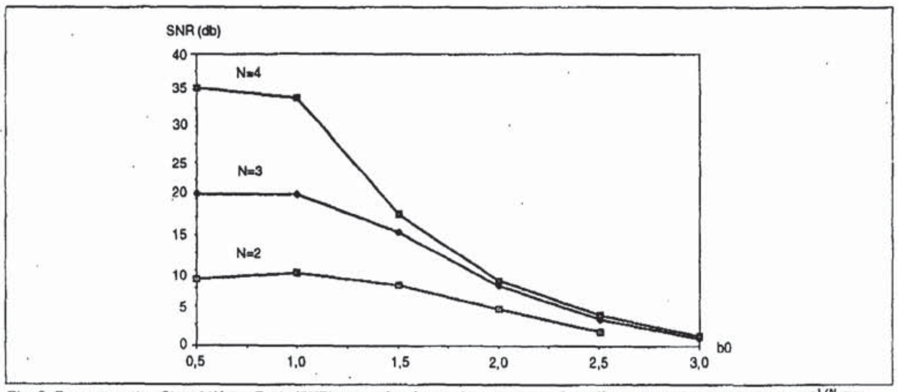


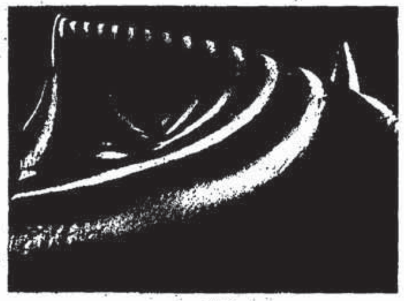
Fig. 8. Reconstruction Signal/Noise Ratio (SNR) error after frame decomposition for different sampling densities $a_0 = 2^{1/N}$ (N = number of voices per octave), $b = a_0^j\, k\, b_0$ (after [DAU90a]). The basic wavelet is the Morlet wavelet (modulated Gaussian) used in [GRO89]. The reconstruction is done "as if" wavelets were orthogonal (see text), and its accuracy grows as N increases and b_0 decreases, i.e. as the density of the sampling grid of Fig. 7 increases. Therefore, redundancy refines the "orthogonal-like" reconstruction.

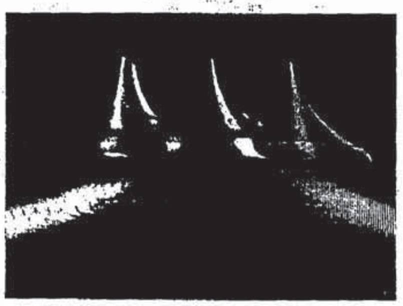
Three Dimensional Displays of Complex Wavelet Transforms

As seen in Box 3, the wavelet transform using a complex wavelet like the Morlet wavelet (a complex sinusoid windowed by a Gaussian) leads to a complex valued function on the plane.

Phase information is also useful and thus, there is interest in a common display of magnitude and phase. This is possible by using height as magnitude and color as phase, leading to so-called "phasemagrams".

Two examples are shown here: a synthetic chirp in the upper figure (similar to the one in Fig. 5.3); and a triangle function below. In both cases, the discontinuous points are clearly identified at small scales (top of the figure). The chirp has two such points (beginning and end), while the triangle has three. At large scales, these signals look just like a single discontinuity, which is what an observer would indeed see from very far away. For the chirp, the phase cycles with increasing speed, as expected.





Signal analyses with a Morlet wavelet. The display shows magnitude as height and phase as color (phasemagram). The horizontal axis is time. Above) a synthetic chirp signal, with frequency increasing with time. Below) a triangle function.

case and show how to construct orthonormal bases of wavelets for continuous-time signals [DAU88].

In the discrete time case, two methods were developed independently in the late seventies and early eighties which lead naturally to discrete wavelet transforms, namely subband coding [CRI76], [CRO76], [EST77] and pyramidal coding or multiresolution signal analysis [BUR83]. The methods were proposed for coding, and thus, the notion of critical sampling (of requiring a minimum number of samples) was of importance. Pyramid coding actually uses some oversampling, but because it has an easier intuitive explanation, we describe it first.

While the discrete-time case has been thoroughly studied in the filter bank literature in terms of frequency bands (see e.g. [VAI87]), we insist here on notions which are closer to the wavelet point of view, namely those of *scale* and *resolution*. Scale is related to the size of the signal, while resolution is linked to the amount of detail present in the signal (see Box 1 and Fig. 9).

Note that the *scale* parameter in discrete wavelet analysis is to be understood as follows: For large scales, dilated wavelets take "global views" of a *subsampled* signal, while for small scales, contracted wavelets analyze small "details" in the signal.

The Multiresolution Pyramid

Given an original sequence $x(n)$, $n \in \mathbf{Z}$, we derive a lower resolution signal by lowpass filtering with a half-band low-pass filter having impulse response $g(n)$. Following Nyquist's rule, we can subsample by two (drop every other sample), thus doubling the scale in the analysis. This results in a signal $y(n)$ given by

$$y(n) = \sum_{k=-\infty}^{+\infty} g(k)\, x(2n-k)$$

The resolution change is obtained by the lowpass filter (loss of high frequency detail). The scale change is due to the subsampling by two, since a shift by two in the original signal $x(n)$ results in a shift by one in $y(n)$.

Now, based on this lowpass and subsampled version of $x(n)$, we try to find an approximation, $a(n)$, to the original. This is done by first upsampling $y(n)$ by two (that is, inserting a zero between every sample) since we need a signal at the original scale for comparison.

$$y'(2n) = y(n), \quad y'(2n+1) = 0$$

Then, $y'(n)$ is interpolated with a filter with impulse response $g'(n)$ to obtain the approximation $a(n)$,

$$a(n) = \sum_{k=-\infty}^{\infty} g'(k)\, y'(n-k)$$

Note that if $g(n)$ and $g'(n)$ were perfect halfband filters (having a frequency passband equal to 1 over the normalized frequency range $-\pi/2$, $\pi/2$ and equal to 0 elsewhere), then the Fourier transform of a(n) would be

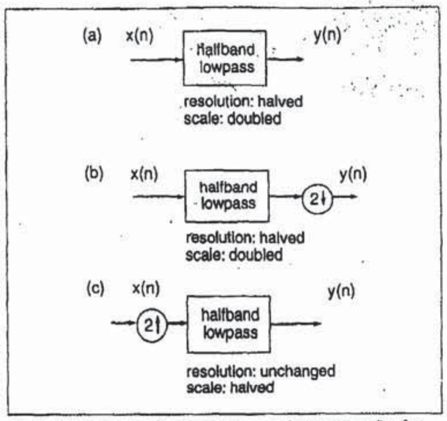


Fig. 9. Resolution and scale changes in discrete time (by factors of 2). Note that the scale of signals is defined as in geographical maps. (a) Halfband lowpass filtering reduces the resolution by 2 (scale is unchanged). (b) Halfband lowpass filtering followed by subsampling by 2 doubles the scale (and halves the resolution as in (a)). (c) Upsampling by 2 followed by halfband lowpass filtering halves the scale (resolution is unchanged).

equal to the Fourier transform of $x(n)$ over the frequency range $(-\pi/2, \pi/2)$ while being equal to zero elsewhere. That is, $a(n)$ would be a perfect halfband lowpass approximation to $x(n)$.

Of course, in general, $a(n)$ is not going to be equal to $x(n)$ (in the previous example, $x(n)$ would have to be a halfband signal). Therefore, we compute the difference between $a(n)$ (our approximation based on $y(n)$) and $x(n)$,

$$d(n) = x(n) - a(n)$$

It is obvious that $x(n)$ can be reconstructed by adding $d(n)$ and $a(n)$, and the whole process is shown in Fig. 10. However, there has to be some redundancy, since a signal with sampling rate f_s is mapped into two signals $d(n)$ and $y(n)$ with sampling rates f_s and $f_s/2$, respectively.

In the case of a perfect halfband lowpass filter, it is clear that $d(n)$ contains exactly the frequencies above $\pi/2$ of $x(n)$, and thus, $d(n)$ can be subsampled by two as well without loss of information. This hints at the fact that critically sampled schemes must exist.

The separation of the original signal $x(n)$ into a coarse approximation $a(n)$ plus some additional detail contained in $d(n)$ is conceptually important. Because of the resolution change involved (lowpass filtering followed by subsampling by two produces a signal with half the resolution and at twice the scale of the original), the above method and related ones are part of what is called Multiresolution Signal Analysis [ROS84] in computer vision.

The scheme can be iterated on $y(n)$, creating a hierarchy of lower resolution signals at lower scales. Because of that hierarchy and the fact that signals become shorter and shorter (or images become smaller and smaller), such schemes are called signal or image pyramids [BUR83].

Subband Coding Schemes

We have seen that the above system creates a redundant set of samples. More precisely, one stage of a pyramid decomposition leads to both a half rate low resolution signal and a full rate difference signal, resulting in an increase in the number of samples by 50%. This oversampling can be avoided if the filters $g(n)$ and $g'(n)$ meet certain conditions [VET90b].

We now look at a different scheme instead, where no such redundancy appears. It is the so-called subband coding scheme first popularized in speech compression [CRI76, CRO76, EST77]. The lowpass, subsampled approximation is obtained exactly as explained above, but, instead of a difference signal, we compute the "added detail" as a highpass filtered version of $x(n)$ (using a filter with impulse response $h(n)$), followed by subsampling by two. Intuitively, it is clear that the "added detail" to the lowpass approximation has to be a highpass signal, and it is obvious that if $g(n)$ is an ideal halfband lowpass filter, then an ideal halfband highpass filter $h(n)$ will lead to a perfect representation of the original signal into two subsampled versions.

This is exactly one step of a wavelet decomposition using $\sin(x)/x$ filters, since the original signal is mapped into a lowpass approximation (at twice the s[illegible]) and

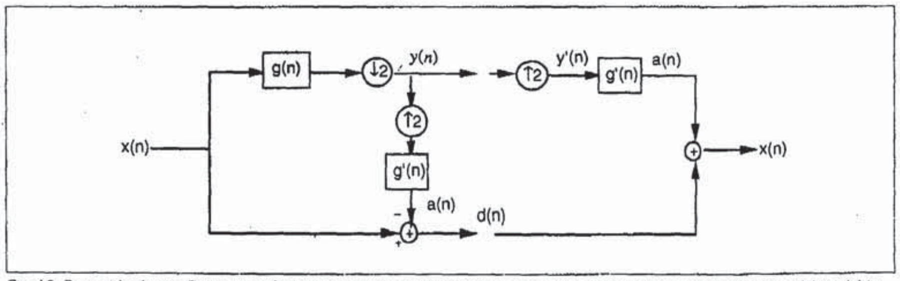


Fig. 10. Pyramid scheme. Derivation of a lowpass, subsampled approximation $y(n)$, from which an approximation $a(n)$ to $x(n)$ is derived by upsampling and interpolation. Then, the difference between the approximation $a(n)$ and the original $x(n)$ is computed as $d(n)$. Perfect reconstruction is simply obtained by adding $a(n)$ back.

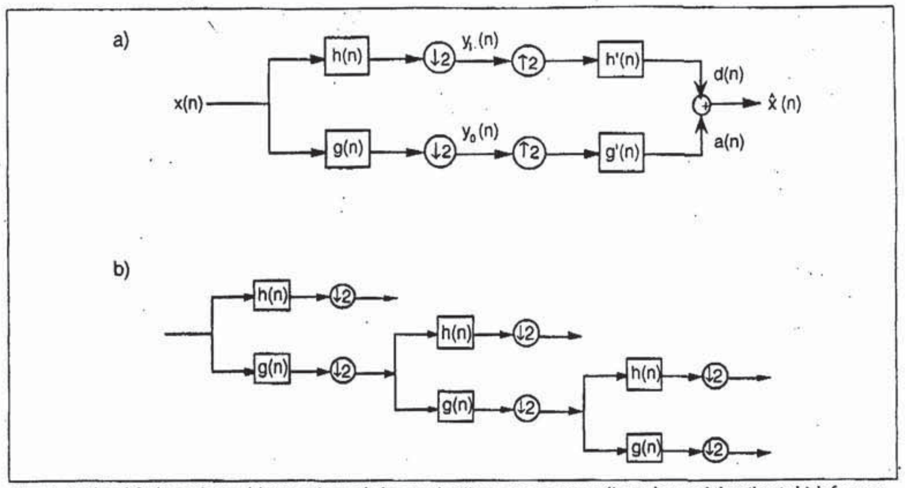


Fig. 11. Subband Coding scheme. (a) Two subsampled approximations, one corresponding to low and the other to high frequencies, are computed. The reconstructed signal is obtained by re-interpolating the approximations and summing them. The filters on the left form an analysis filter bank, while on the right is a synthesis filter bank. (b) Block diagram (Filter Bank tree) of the Discrete Wavelet Transform implemented with discrete-time filters and subsampling by two. The frequency resolution is given in Fig. 3b.

an added detail signal (also at twice the scale). In particular, using these ideal filters, the discrete version is identical to the continuous wavelet transform.

What is more interesting is that it is not necessary to use ideal (that is, impractical) filters, and yet $x(n)$ can be recovered from its two filtered and subsampled versions which we now call $y_0(n)$ and $y_1(n)$. To do so, both are upsampled and filtered by $g'(n)$ and $h'(n)$ respectively, and finally added together, as shown in Fig. 11a. Now, unlike the pyramid case, the reconstructed signal (which we now call $\hat{x}(n)$) is not identical to $x(n)$, unless the filters meet some specific constraints. Filters that meet these constraints are said to have *perfect reconstruction* property, and there are a number of papers investigating the design of perfect reconstruction filter banks [MIN85, SMI86, VAI88, VET86].

The easiest case to analyze appears when the analysis and synthesis filters in Fig. 11a are identical (within time-reversal) and when perfect reconstruction is achieved (that is, $\hat{x}(n) = x(n)$, within a possible shift). Then it can be shown that the subband analysis/synthesis corresponds to a decomposition onto an orthonormal basis, followed by a reconstruction which amounts to summing up the orthogonal projections. We will assume FIR filters in the following. Then, it turns out that the highpass and lowpass filters are related by

$$h(L-1-n) = (-1)^n\, g(n) \qquad (12)$$

where L is the filter length (which has to be even). Note that the modulation by $(-1)^n$ transforms indeed the lowpass filter into a highpass one.

Now, the filter bank in Fig. 11a, which computes convolutions followed by subsampling by two, evaluates inner products of the sequence $x(n)$ and the sequences $\{g(-n+2k),\ h(-n+2l)\}$ (the time reversal comes from the convolution, which reverses one of the sequences). Thus

$$y_0(k) = \sum_n x(n)\, g(-n+2k)$$
$$y_1(k) = \sum_n x(n)\, h(-n+2k)$$

Because the filter impulse responses form an orthonormal set, it is very simple to reconstruct $x(n)$ as

$$x(n) = \sum_{k=-\infty}^{\infty} \left[\, y_0(k)\, g(-n+2k) + y_1(k)\, h(-n+2k) \right] \qquad (13)$$

that is, as a weighted sum of the orthogonal impulse responses, where the weights are the inner products of the signal with the impulse responses. This is of course the standard expansion of a signal into an orthonormal basis, where the resynthesis is the sum of the orthogonal projections (see *Introduction to orthogonal wavelet bases* above).

From (12), (13) it is also clear that the synthesis filters are identical to the analysis filters within time reversal.

Such orthogonal perfect reconstruction filter banks have been studied in the digital signal processing literature, and the orthonormal decomposition we just indicated is usually referred to as a "paraunitary" or "lossless" filter bank [VAI89]. An interesting property of such filter banks is that they can be written in lattice form [VAI88], and that the structure and properties can be extended to more than two channels [VAI87, VAI89, VET89]. More general perfect reconstruction (biorthogonal) filter banks have also been studied (see e.g. [VET86, VET90b, COH90a]). It has been also noticed [MAL89b, SHE90, RIO90b] that filter banks arise naturally when implementing the CWT.

Note that we have assumed linear processing throughout. If non-linear processing is involved (like quantization), the oversampled nature of the pyramid scheme described in the preceding section may actually lead to greater robustness.

The Discrete Wavelet Transform

We have shown how to decompose a sequence $x(n)$ into two subsequences at half rate, or half resolution, and this by means of "orthogonal" filters (orthogonal with respect to even shifts). Obviously, this process can be iterated on either or both subsequences. In particular, to achieve finer frequency resolution at lower frequencies (as obtained in the continuous wavelet transform), we iterate the scheme on the lower band only. If $g(n)$ is a good halfband lowpass filter, $h(n)$ is a good halfband highpass filter by (12). Then, one iteration of the scheme on the first lowband creates a new lowband that corresponds to the lower quarter of the frequency spectrum. Each further iteration halves the width of the lowband (increases its frequency resolution by two), but due to the subsampling by two, its time resolution is halved as well. At each iteration, the current high band portion corresponds to the difference between the previous lowband portion and the current one, that is, a passband. Schematically, this is equivalent to Fig. 11b, and the frequency resolution is as in Fig. 3b.

An important feature of this discrete algorithm is its relatively low complexity. Actually, the following somewhat surprising result holds: independent of the depth of the tree in Fig. 11b, the complexity is linear in the number of input samples, with a constant factor that depends on the length of the filter. The proof is straightforward. Assume the computation of the first filter bank requires C_0 operations per input sample (C_0 is typically of the order of L). Then, the second stage requires also C_0 operations per sample of its input, but, because of the subsampling by two, this amounts to $C_0/2$ operations per sample of the input signal. Therefore, the total complexity is bounded by

$$C_{total} = C_0 + \frac{C_0}{2} + \frac{C_0}{4} + \ldots < 2C_0$$

which demonstrates the efficiency of the discrete wavelet transform algorithm and shows that it is independent of the number of octaves that one computes. This bounded complexity had been noticed in the multirate filtering context [RAM88]. Further developments can be found in [RIO91a]. Note that a possible drawback is that the delay associated with such an iterated filter bank grows exponentially with the number of stages.

Iterated Filters and Regularity

There is a major difference between the discrete scheme we have just seen and the continuous time wavelet transform. In the discrete time case, the role of the wavelet is played by the highpass filter $h(n)$ and the cascade of subsampled lowpass filters followed by a highpass filter (which amounts to a bandpass filter). These filters, which correspond roughly to octave band filters, unlike in the continuous wavelet transform, are not exact scaled versions of each other. In particular, since we are in discrete time, scaling is not as easily defined, since it involves interpolation as well as time expansion.

Nonetheless, under certain conditions, the discrete system converges (after a certain number of iterations) to a system where subsequent filters are scaled versions of each other. Actually, this convergence is the basis for the construction of continuous time compactly supported wavelet bases [DAU88].

Now, we would like to find the equivalent filter that corresponds to the lower branch in Fig. 11b, that is the iterated lowpass filter. It will be convenient to use z-transforms of filters, e.g. $G(z) = \sum_n g(n)\, z^{-n}$ in the following. It can be easily checked that subsampling by two followed by filtering with $G(z)$ is equivalent to filtering with $G(z^2)$ followed by the subsampling (z^2 inserts zeros between samples of the impulse response, which are removed by the subsequent subsampling). That is, the first two steps of lowpass filtering can be replaced by a filter with z-transform $G(z)\cdot G(z^2)$, followed by subsampling by 4. More generally, calling $G^i(z)$ the equivalent filter to i stages of lowpass filtering and subsampling by two (that is, a total subsampling by 2^i), we obtain

$$G^i(z) = \prod_{l=0}^{i-1} G(z^{2^l}) \tag{14}$$

Call its impulse response $g^i(n)$.

As i infinitely increases, this filter becomes infinitely long. Instead, consider a function $f^i(x)$ which is piecewise constant on intervals of length $1/2^i$ and has value $2^{i/2}\, g^i(n)$ in the interval $[n/2^i, (n+1)/2^i]$. That is, $f^i(x)$ is a staircase function with the value given by the samples of $g^i(n)$ and intervals which decrease as 2^{-i}. It can be verified that the function is supported on the interval $[0, L-1]$, where L is the length of the filter $g(n)$. Now, for i going to infinity, $f^i(x)$ can converge to a continuous function $g_c(x)$, or a function with finitely many discontinuities, even a fractal function, or not converge at all (see Box 5).

A necessary condition for the iterated functions to converge to a continuous limit is that the filter $G(z)$ should have a sufficient number of zeros at $z = -1$, or half sampling frequency, so as to attenuate repeat spectra [DAU88, DAU90b, RIO91b]. Using this condition, one can construct filters which are both orthogonal and converge to continuous functions with compact support. Such filters are called *regular*, and examples can be found in [DAU88, COH90a, DAU90b, RIO90b, VET90b]. Note that the above condition can be interpreted as a *flatness* condition on the spectrum of $G(z)$ at half sampling frequency. In fact, it can be shown

Box 5: Regular Scaling Filters

It is well known that the structure of computations in a Discrete Wavelet Transform and in an octave-band filter bank are identical. Therefore, besides the different views and interpretations that have been given to them, the main difference lies in the filter design. Wavelet filters are chosen so as to be *regular*. Recall that this means (with the same notation as used in the main text sections on iterated filters), that the piecewise constant function associated with the discrete wavelet sequence $h_J(n)$ of z-transform $G^J(z)H(z^{2^J})$ converges (e.g. pointwise), as J indefinitely increases, to a regular limit function $h_c(x)$. Equivalently, the piecewise constant function associated with the discrete "scaling" sequence $g_J(n)$ of z-transform $G^J(z)$ converges to a regular limit function $g_c(x)$. By "regular" we mean that the continuous-time wavelet $h_c(x)$ (or the scaling function $g_c(x)$) is at least continuous, or better, once or twice continuously differentiable. The regularity order is the number of times $h_c(x)$ (or $g_c(x)$) is continuously differentiable. Figures 12a and 12b show two examples, one where $g_c(x)$ is almost three times continuously differentiable and another where $g_J(n)$ diverges with fractal behavior.

Note that there are a number of classical filters, designed for two-band filter banks, which, unlike wavelet filters, are not regular. Figures 12c through 12f show two well-known examples: a Johnston filter [JOH80], and a Smith and Barnwell filter [SMI86]. The latter allows perfect reconstruction, while the former does not. Figure 12d shows that the Smith and Barnwell discrete sequences $h_J(n)$ do not tend to regular limit functions, but rather diverge. This is not surprising since the necessary condition that the low-pass filter has a zero at half the sampling frequency is violated (although this filter has 40 dB attenuation in the stop band [SMI86]). This eventually results, when J increases, in small, but rapid oscillations in $h_J(n)$. As for the Johnston filter (Figs. 12e and 12f), it can be shown that the wavelet limit function is continuous but not differentiable.

For wavelet filters, the more regular the limit function, the faster the convergence to this limit [RIO91b] — and in practice the convergence is very fast. This justifies the study of the limit $h_c(x)$, which is almost attained after a few octaves of a logarithmic decomposition. Since an error in a wavelet coefficient (due e.g. to quantization) results, after reconstruction, in an overall error proportional to a discrete wavelet $h_J(n)$, regularity seems a nice property, e.g., to avoid visible distortion on a reconstructed image [ANI90].

From equations (12), (15) and (16), the knowledge of $g(n)$ suffices to determine the limit $h_c(x)$. Several methods have been developed to estimate the regularity order of $h_c(x)$ from the coefficients $g(n)$. Most are based on Fourier transform techniques [DAU88, COH90b]. Recently, time-domain techniques have been developed which provide optimal estimates [DAU90c, RIO91b].

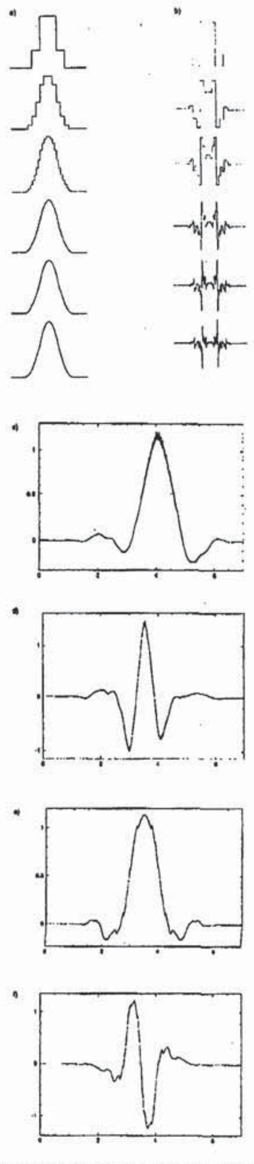


Fig. 12. (a) Iterated low-pass filter $g_J(n)$ with $g(n) = (1,3,3,1)$, converges to a regular, smooth function. (b) Iterated low-pass filter with $g(n) = (-1,3,3,-1)$ converges to a fractal function (see text). (c) Smith and Barnwell 8-tap lowpass filter [SMI86], iterated 8 times. Divergence occurs, due to rapid oscillations in the temporal waveform. (d) Corresponding continuous-time wavelet (after [RIO90b]). (e) Johnston 8-tap lowpass filter [JOH80], iterated 8 times. The limit function is not differentiable. (f) Corresponding continuous-time wavelet.

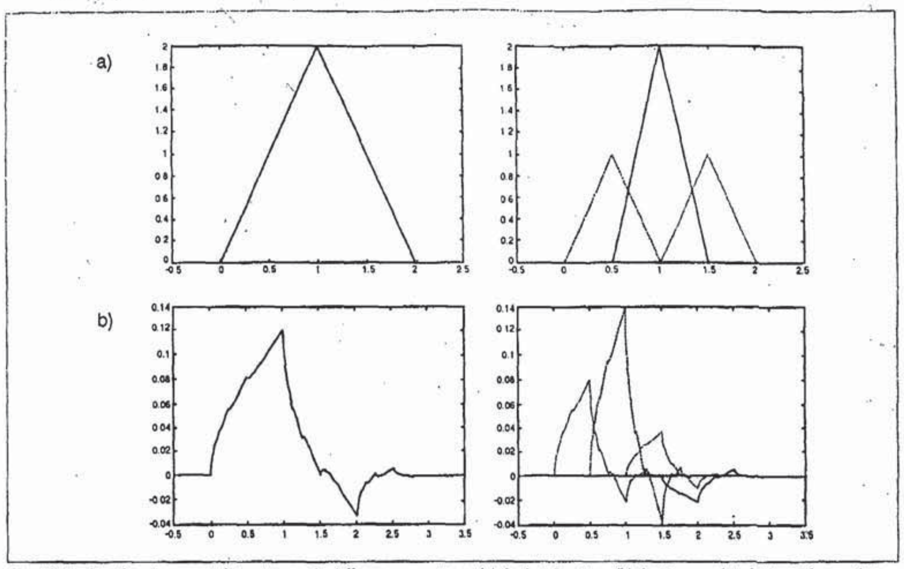


Fig. 13. Scaling functions satisfying two-scale difference equations. (a) the hat function. (b) the D_4 wavelet obtained from a 4-tap regular filter by Daubechies.

[AKA90], [SHE90] that the well-known Daubechies orthonormal filters [DAU88] are deduced from "maximally flat" low-pass filters [HER71]. Note that there are many other choices that behave very differently in terms of phase, selectivity in frequency, and other criteria (see e.g. [DAU90b]). An important issue related to regular filter design is the derivation of simple estimates for the regularity order (see Box 5).

It is still not clear whether regular filters are most adapted to coding schemes [ANI90]. The minimal regularity order necessary for good coding performance of discrete wavelet transform schemes, if needed at all, is also not known and remains a topic for future investigation.

Scaling Functions and Wavelets Obtained from Iterated Filters

Recall that $g_c(x)$ is the final function to which $f^i(x)$ converges. Because it is the product of lowpass filters, the final function is itself lowpass and is called a "*scaling function*" because it is used to go from a fine scale to a coarser scale. Because of the product (14) from which the scaling function is derived, $g_c(x)$ satisfies the following two scale difference equation [DAU90c]:

$$g_c(x) = \sum_{n=-\infty}^{\infty} g(n)\, g_c(2x-n) \qquad (15)$$

Figure 13 shows two such examples. The second one is based on the 4-tap Daubechies filter which is regular and orthogonal to its even translates [DAU88].

So far, we have only discussed the iterated lowpass and its associated scaling function. However, from Fig. 11b, it is clear that a bandpass filter is obtained in the same way, except for a final highpass filter. Therefore, in a fashion similar to (15), the wavelet $h_c(x)$ is obtained as

$$h_c(x) = \sum_{n=-\infty}^{\infty} h(n)\, g_c(2x-n) \qquad (16)$$

that is, it also satisfies a two scale equation.

Now, if the filters $h(n)$ and $g(n)$ form an orthonormal set with respect to even shifts, then the functions $g_c(x-l)$ and $h_c(x-k)$ form an orthonormal set (see Box 6). Because they also satisfy two scale difference equations, it can be shown [DAU88] that the set $h_c(2^{-l}x-k)$, $l, k \in \mathbb{Z}$, forms an orthonormal basis for the set of square integrable functions.

Figure 14 shows two scales and shifts of the 4-tap Daubechies wavelet [DAU88]. While it might not be obvious from the figure, these functions are orthogonal to each other, and together with all scaled and translated versions, they form an orthonormal basis.

Figure 15 shows an orthogonal wavelet based on a length-18 regular filter. It is obviously a much smoother function (actually, it possesses 3 continuous derivatives).

Finally, Fig. 16 shows a biorthogonal set of linear phase wavelets, where the analysis wavelets are orthogonal to the synthesis wavelets. These were obtained from a biorthogonal linear phase filter bank with length-18 regular filters [VET90a, VET90b].

We have shown how regular filters can be used to