Applications for data hiding

by W. Bender
W. Butera
D. Gruhl
R. Hwang
F. J. Paiz
S. Pogreb

In an earlier paper, "Techniques for Data Hiding," the overall goals and constraints of information-hiding problem space and a variety of approaches to information hiding in image, audio, and text were described. In this sequel, information-hiding goals and applications are expanded beyond watermarking to encompass the more general concept of information tagging. As a framework for illustrating these concepts, multiple-bit and midlevel image-representation information-hiding techniques are described. A range of applications, including anticounterfeiting and authentication, is explored.

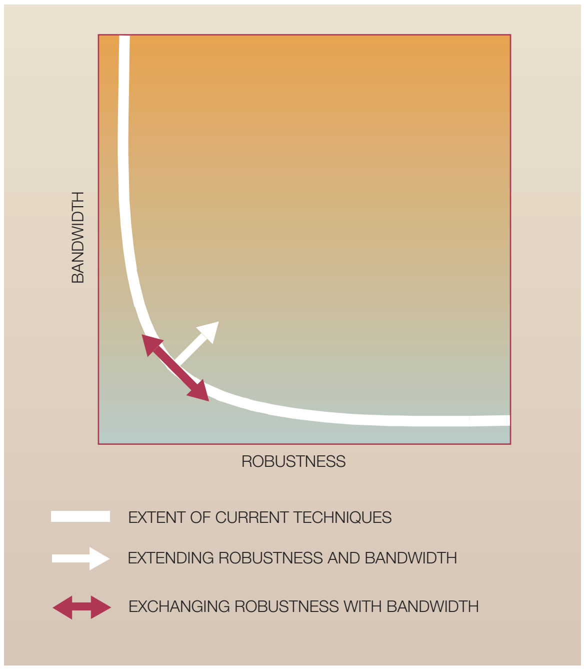
Information (or data) hiding, a form of steganography, embeds data into digital media for the purpose of identification, annotation, and copyright. (As defined at the first international workshop on information hiding,[1] cryptography protects the content of messages while steganography conceals their very existence.) Several constraints affect this process: the quantity of data to be hidden, the need for invariance of these data under distortion of the cover signal, and the degree to which the data must be immune to interception, modification, or removal by a third party.

In "Techniques for Data Hiding"[2] the information-hiding problem space was characterized by a tradeoff between robustness and bandwidth. By constraining the degree of cover-signal degradation, an information-hiding method can operate with either high embedded data rate or high resistance to modification, but not both. As one increases, the other must decrease.

Trends such as the growing popularity of MPEG (Motion Picture Experts Group) Audio Layer 3 (MP3) have created a climate of corporate vigilance regarding the protection of intellectual property. This climate has spawned a corresponding digital-watermarking imperative, driving the engineering community to focus on one segment of the information-hiding problem space—developing more secure methods of embedding watermarks in sound, image, and video files. The metric of success driving this effort is resistance to attack. This narrow focus on watermarking has an analogy in the development of digital-television systems in the 1980s. The research community defined the problem space in terms of efficient use of bandwidth at the expense of consideration of "out-of-topic" features such as scalability that broaden the potential application domain.

Some exceptions to this focus on watermarking include Anderson and Lee's Jikzi system,[3] which addresses secure document publication as opposed to copy control. Other researchers, such as Kundur and Hatzinakos[4] and Quelez,[5] are examining the problem of image authentication. And a few researchers are exploring information hiding within the context of privacy. For example, Demuth and Rieke's JANUS system[6] uses information hiding to provide anonymity for content providers on the World Wide Web. Anderson et al.'s steganographic file system[7] is an-

Figure 1 Conceptual information-hiding problem space


BANDWIDTH
ROBUSTNESS
EXTENT OF CURRENT TECHNIQUES
EXTENDING ROBUSTNESS AND BANDWIDTH
EXCHANGING ROBUSTNESS WITH BANDWIDTH

other example of an unexpected, but practical application of information hiding.

There are some efforts to use information hiding to create general-purpose data channels. Gerasimov and Bender[8] use information hiding as a mechanism for device-to-device communication. Adelson[9] uses information hiding as a means of embedding arbitrary meta-data into images. Schreiber et al.[10] use information hiding to embed alternative channels into television broadcasts. But the volume of research on watermarking, e.g., hidden copyright messages, is an order of magnitude greater than that of other applications of information hiding, as reflected in the literature and in compilations such as Anderson and Petitcolas' annotated bibliography on information hiding.[11] (The reader interested in the latest trends in watermarking should consult References 11–16.)

An expanded information-hiding problem space should be characterized as a general-purpose data channel. There are three attributes of this channel that expand the potential application domain: (1) techniques that combine *strong* and *weak* data embedding offer a flexibility that gives application designers the ability to move fluidly between robustness and bandwidth considerations; (2) embedding meta-data leads to improved management of assets, not just for the purposes of intellectual-property protection, but for general process and application control; and (3) in a network computing environment, a small amount of embedded data is not a significant limitation. Rather than struggling to embed all of the information that one would like to associate with a particular image, a URL (uniform resource locator), filename, or unique serial number can be embedded, using off-image storage for large amounts or dynamic updating of information over the network.

In this paper, information hiding is explored in light of several approaches, multiple-layer and midlevel vision techniques, and several application domains, transfer of embedded data to and from nondigital (physical) media and the authentication and characterization of images. First, we describe a multiple-layer technique as an exemplar of flexible encoding. This example is then used to explore two applications that exploit information hiding, anticounterfeiting and authentication. Finally, we describe a midlevel vision technique that, although not yet mature enough to have applicability beyond authentication, achieves data embedding outside of the usual considerations of signal-to-noise ratio.

Multiple-layer encoding

Our interest in multiple-layer techniques lies in their ability for exchanging robustness with bandwidth, an orthogonal goal from watermarking (see Figure 1). The notion of layering multiple watermarks into a cover signal is not unprecedented. For example, Mintzer and Braudaway[17] analyze the order in which watermarks are applied, based upon their fragility, and Fridrich and Goljan[18] have described a method to convert a multiple-bit technique into a one-bit technique and *vice versa*. The purpose of the discussion below is to provide an example of a multiple-layer-encoding technique that is used as the basis of a later discussion of information-hiding applications.

Patchwork (described in detail in Bender et al.[2]) is an information-hiding technique that uses a statistical analysis to detect a single, specific bit in an image. Patchwork imperceptibly embeds in a host image a specific statistic that has a Gaussian distribution. The basic algorithm is to modify the original picture by choosing a pseudorandom path of length n through the image and modifying "patches" centered at sequential pairs of points. By raising the brightness in one patch and lowering it in the other by δ, the *expected* value of the sum of the differences between patch pairs is modified. This deviation in expected value, S_n, is usually interpreted as a watermark (see Figure 2).

Patchtrack[19] is a variant of the Patchwork algorithm that addresses a problem inherent in information-hiding techniques for digital images—that of alignment during data extraction. This alignment problem can be solved using a search approach where a combination of coarse orientation-detection, random-search, and gradient-descent methods are employed,[20] the goal being to reduce the resource overhead associated with detection of the stego (hidden) data. When used in conjunction with the Patchwork algorithm, it is possible to create an information-hiding and retrieval system that is robust toward rotation, cropping, and noise; successful data extraction can occur with a high degree of certainty from scanned images, rotated images, and partially obscured images. (For an example of another approach to affine resilience, see Pereira and Pun.[21])

Both Patchwork and Patchtrack are amenable to multiple-layer extensions. A multiple-bit encoder is a technique for embedding an ordered sequence of bits. Multiple-layer encoding is implemented by first encoding a *strong* bit (large pseudorandom path length n_0) using the Patchwork method. Additional bits are then encoded weakly (small pseudorandom path length n_i) with each successive path representing one bit. A one or a zero is encoded by either a positive or negative S_{n_i} (the basic Patchwork encoding algorithm has the ability to encode with both positive and negative δs). After orienting about the strong bit, it is only necessary to identify the sign of the following bits. A positive S_{n_i} represents a 1 while a negative S_{n_i} represents a 0. Data are decoded by stepping through all of the keys and calculating the corresponding S_{n_i}.

Each bit extracted from the stego image has an associated certainty. By adjusting path lengths and patch depths, these statistics can be adjusted. Multiple-bit encoding uses ECC (error-control coding) to adjust decoding accuracy relative to the magnitude of δ. An ECC scheme can be used in conjunction with decreasing δ (and consequently decreasing certainty of the embedded data). Or redundant coding could be used to ensure a higher degree of certainty at a fixed δ. An example of multiple-bit error-control coding is shown in Table 1. (An interesting subject for further study is the design of ECC methods that are optimized for variable but known certainty associated with individual bits.)

Figure 2 **A schematic of multiple iterations through the Patchwork method. At each iteration, two patches (*A_i* and *B_i*) are chosen pseudorandomly in the image. *A_i* is lightened by δ while *B_i* is darkened by δ.**

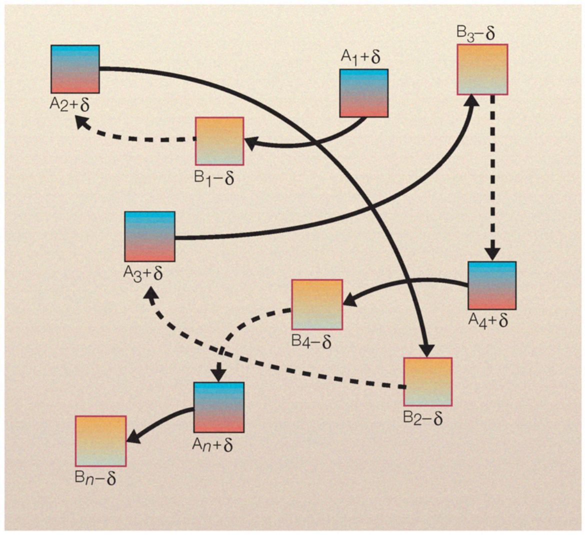


Table 1 S_{n_i} values under ECC scheme

Seed	Decode Value (S_n)	Decoded Bit
Strong bit	39896	—
1	−3470	0
2	−3154	0
3	−13794	0
4	7322	1
5	2588	1
6	5894	1
7	9762	1
8	5252	1
9	7076	1
10	−7950	0
11	−11270	0
12	−8568	0

The quantity of data that can be stored by the multiple-bit encoding described above depends on the frequency content, the size of the image, and the desired robustness of the encoding. In a typical 640 × 480 continuous-tone image, this value ranges from 64 to 256 bits prior to any postprocessing for error correction. This small amount of data is not a significant limitation given the availability of network access. Rather than embedding all of the information that one would like to associate with a particular image, a URL, filename, or unique serial number can be embedded, allowing for the off-image storage of dynamic information.

One way to put off-image storage into practice would be to implement it as a service on the Internet. A client application can be used to extract the database pointer. A 64-bit database pointer can be broken down into a 32-bit Internet Protocol (IP) address and a 32-bit pointer sent in a single User Datagram Protocol (UDP) packet to a service bureau. UDP has minimal overhead since it is connectionless. Data are sent without any overhead of negotiation.

The following two sections detail two applications of multiple-layer encoding and off-image storage: (1) anticounterfeiting exploits multiple-bit encoding in order to exchange data between virtual and physical instances of an image; and (2) tunable image characterization (authentication) exploits multiple-layer encoding to embed both database pointers and markers in images.

Anticounterfeiting

Recent advances in the technology for ink-jet printers and scanners have created a new way for computer users to inexpensively create high-quality color reproductions of any original document. One obvious problem that arises from these advances is the ability to casually create counterfeit currency (or some other valuable document) that looks real enough to fool anyone who does not inspect it carefully. The embedding of a digital signature into a secure document that can be recognized by a printer as it prints the document is a potential deterrent to casual counterfeiters. If a secure document is recognized, the printer can refuse to complete the print job and issue an appropriate warning. For this deterrent to serve as a viable solution, however, it must provide a high level of security while requiring only a minimal addition to a printer. Potentially, such anticounterfeiting methods could be used to protect currency, stock certificates, bank checks, or airline tickets from illicit reproduction. These methods are of particular applicability when unsuspecting individuals are exchanging documents for something of value.

To serve as an example, the implementation of an encoding and detecting algorithm, Tartan Threads,[20,22] based upon multiple-layer encoding, is detailed and evaluated regarding its ability to address the casual counterfeiting problem. (The name Tartan Threads was chosen because the method uses striped patterns that are reminiscent of the security threads found in U.S. paper currency.) Tartan Threads is designed to hold information in a fixed-size linearly contiguous space to allow for time-efficient decoding with a high degree of certainty. When used in conjunction with a system that marks continuous-tone printouts with a unique digital signature, it provides ink-jet printers copy protection commensurate with color copiers.

Marvel et al.[23] have implemented a blind digital steganography system called Spread Spectrum Image Steganography (SSIS) built upon a two-dimensional spread-spectrum method. Through the use of image restoration techniques, an estimate of the original image is recreated and then subtracted from the encoded document to reveal the encoded information. As compared to Tartan Threads, SSIS yields a higher encoding bandwidth and a lower perceptibility. The encoding, however, is not intended to survive a printing and scanning image cycle, nor is it amenable to quick decoding.

Herrigel et al.[24] and Fridrich et al.[25] both describe image watermarking methods built on spread-spectrum techniques combined with a public-key encryption system for authentication of both the author and purchaser of digital images. Herrigel's technique, like Tartan Threads, redundantly encodes several small areas of the image with identical watermarks, although Herrigel et al.'s watermark is in the Fourier transform domain rather than the spatial domain. The technique is intended to survive cropping, as the areas are tiled and encoded in the single orientation. Rotation and scaling transformations are handled through the analysis of these encoded blocks in a polar coordinate space. By calculating the Fourier transform of the entire image, it is possible to characterize any rotation or scaling that had been applied to the image. Fridrich et al. combines global and local encoding schemes. As an additional security measure, encoding patterns are generated using a secret key. Since these techniques involve two-dimensional encoding methods, decoding requires extensive processing times for larger images.

These methods focus primarily on marking images that are intended to be distributed in digital form. They may tolerate some loss but they are not intended to survive the combination of sampling and quantization errors introduced by a printing and scanning cycle, the nonlinear effects produced in various printing processes, etc. Tartan Threads are intended for images that will be distributed in printed form. The encoding survives even at low scanning resolutions (100 dpi [dots per inch]) and can be decoded without complicated analysis of the encoded image—only a small contiguous area of the image needs to be processed.

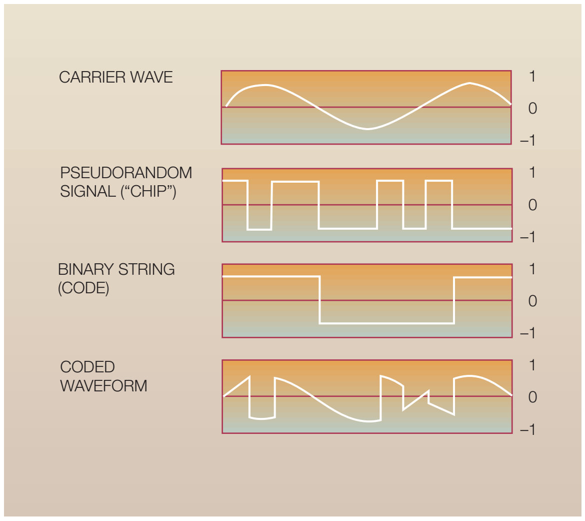
Any encoding on an actively circulated document, such as currency, must be detectable even after wear from usage. (An interesting area for further work would be to create a model for how such documents typically wear over time—it is unknown if such a model publicly exists.) Since different users—with different equipment—will potentially be scanning the protected document, the encoding must also survive any nongeometric transformations and lossy image transformations that may result from the use of different image file formats, compression methods, slight rotations, imperfect color sampling, and resampling at varying resolutions and offsets. Ideally, the encoding would also survive any transformation a user is likely to apply to the digitized image data that does not call attention to the human eye.

Because an ink-jet printer renders images line-by-line and often has only enough memory to buffer a few image lines (typically between 16 kilobytes to 1 megabyte of buffer memory for even the more sophisticated models), all decoding must rely on only a small portion of the document data. Ideally, the decoder would be created with inexpensive hardware capable of affordable searching for encoded information in every print line. Any solution must also have extremely low probabilities of false triggering to prevent disrupting consumers who are using their printers for legitimate purposes.

Encoding. A version of linear direct sequence spread spectrum (DSSS)[26] was chosen as an encoding method that meets the criteria outlined. Traditional linear DSSS involves creating a *carrier wave* of length *data rate* at a known frequency and phase, multiplying it by a *chip* signal and adding it onto a *host* (or cover) signal.

The Tartan Threads method uses a carrier wave that is created in the brightness plane of an image. The carrier wave is summed with a row of pixels of length *data rate* in the cover image. The phase of the carrier wave in this region is set to either 0 or 180 degrees in order to encode a single bit of stego information. The chip signal is a pseudorandomly generated sequence with value of -1 or 1, alternating at a given chip rate. Multiplying the carrier wave by the chip signal spreads the signal so it appears as random noise on the image (see Figure 3).

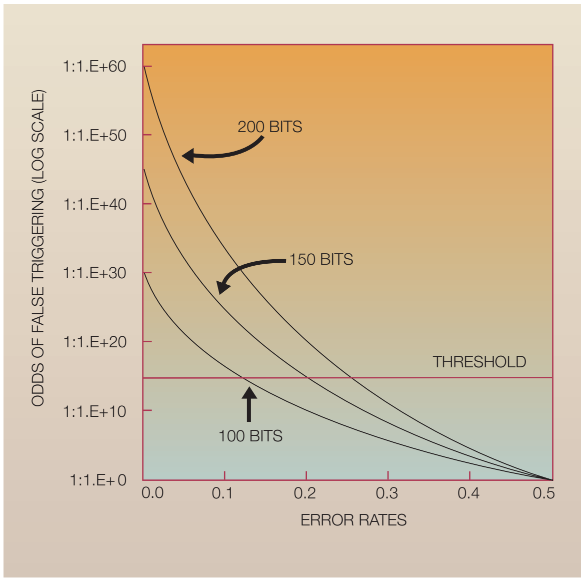
Figure 3 Synthesized spread-spectrum information encoded by the direct-sequence method



Five parameters define Tartan Threads encoding: (1) the data rate, the number of pixels in one line of the image used to encode a single bit; (2) the amplitude of the carrier wave; (3) its frequency, theta, which is measured in number of cycles per data rate; (4) the chip rate; and (5) the spatial frequency of the encoding in the host document, i.e., the dpi at which the host image is encoded. These parameters must be permanently set for decoding, with the exception of the amplitude, which can vary depending on characteristics of the cover image.

The overarching consideration in designing the encoding parameters is to determine how many bits are sufficient to provide certain identification. If the Tartan Threads system is to be installed in all ink-jet printers, it is important that it not prevent consumers from using their printers for a legitimate purpose—the probability of false triggering should be infinitesimal. Assuming that an unencoded image is equally likely to decode a 1 or 0 in any bit position (an assumption that is made all the more reasonable by the fact that a pseudorandom chip signal is used in decoding), false triggering can be analyzed as a series of Bernoulli trials[27] with probability, P,

Figure 4 Odds of false triggering as a function of the number of bits



of 0.5. When attempting to decode an n-bit thread in an unencoded image, the probability of decoding exactly k_0 bits correctly is:

$$P_k(k_0) = \binom{n}{k_0} P^{k_0}(1-P)^{n-k_0} = \binom{n}{k_0}\left(\frac{1}{2}\right)^n \qquad (1)$$

The probability of false triggering with a threshold of k_0 equals the probability of decoding k_0 or more bits correctly, which is:

$$P_{k \geq k_0}(k_0) = \sum_{j=k_0}^{n} \binom{n}{j}\left(\frac{1}{2}\right)^n \qquad (2)$$

Figure 4 shows the probability of false triggering for a given accepted error-rate tolerance for a single thread at levels of 200, 150, or 100 bits of encoding. A threshold for acceptable false triggering probability of approximately 1 in 10^{15} is derived by calculating the number of placement possibilities of a thread while keeping the expected probability of triggering below 1 in 10^9. With a 175-bit thread, a 20.0 percent error rate (35 bit errors) or less is sufficient to show reasonable mathematical certainty that an image is marked.

Robust encoding. Some bandwidth potential of the linear DSSS must be allocated to robustness. In a raster-scanned image, an increased chip length will accommodate for alignment errors along the raster (or horizontal axis). Vertical alignment errors can be accounted for by repeating an identical carrier-signal encoding for a number of raster lines equal to the chip length.

In order to ensure robustness to resampling and lossy image file formats, a low spatial resolution, 200 dpi, is chosen for the encoding. Experiments show that when encoding at this resolution, a chip length of 4 is adequate for accurate sampling. Since carrier waves require two samples per cycle for accurate rendering, theta was chosen simply as one half the number of chips in one data rate (see Figure 5). Choosing an appropriate data rate involves balancing the need for accurately decoding each bit with the amount of certainty provided by the overall encoding. Tables 2 and 3 show experimentally derived error responses for varying data and chip rates. In subsequent experiments, using a 2.24-inch by 0.5-inch space (the width of U.S. bills times the width of a print-head buffer) for each thread, a data rate of 64 pixels was chosen. This rate resulted in bit-error rates of 15 percent or more, but allowed for the encoding of 175 bits of raw information per thread.

The physical size of each thread, when compounded with sparse printing in many of the target documents (currency, for example), leads to limited placement options. Also, the high-amplitude signals needed for the encoding cannot overlap without disrupting one another and rendering their intersection visible. For all of these reasons, the actual number of threads that can be placed into a typical security document is limited.

Visibility masks. To prevent the encoding from becoming too noticeable, a visibility mask is created and used to scale the stego-signal amplitude. Since in the human visual system a signal may be masked by the presence of other high-frequency signals, effective information hiding may utilize the presence of high-frequency areas in the image.[28,29] A visibility mask provides an estimate of contrast sensitivity by plotting the relative amount of high-frequency activity in each region of an image. A simple visibility mask is made by subtracting low-frequency values from an image and rescaling the resultant image from zero to one. When Tartan Threads are embedded in an image, the amplitude of the stego signal is

Table 2 Error rates for varying samples, theta values, chip lengths, and data rate = 256

Samples	Chip Length	Theta	Error Rate (%)
128	2	64	4.58
64	4	32	5.00
32	8	16	11.67
16	16	8	16.67
8	32	4	20.00

Table 3 Error rates for varying samples, theta values, data rates, and chip length = 4

Samples	Data Rate (bits/second)	Theta	Error Rate (%)
128	512	64	2.00
64	256	32	50.00
32	128	16	7.50
16	64	8	15.25
8	32	4	21.38

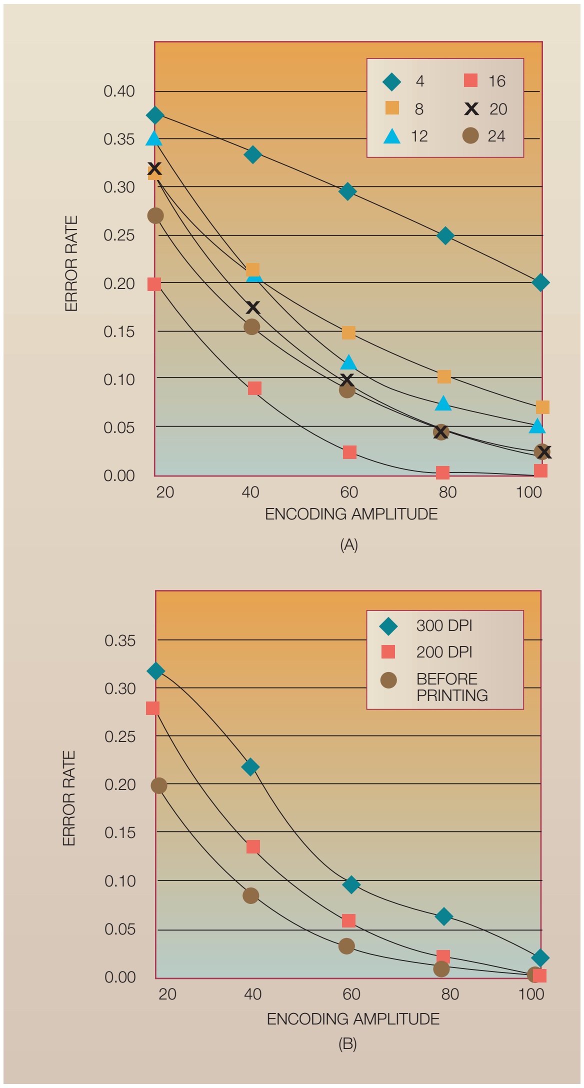
scaled by a corresponding visibility-mask value. This is to ensure strong but "invisible" encoding.

Rotation resistance. In order to be properly decoded, a Tartan Thread has to fit entirely within the print-head buffer. Threads, as described above, can only be decoded in a limited range of orientations, since any small rotation will result in a sampling misalignment. In order to be detected when counterfeiters print at arbitrary orientations, multiple threads are imbedded.

How many threads should be placed in a document to achieve an optimal level of protection? On a very small scale, the modified linear DSSS technique has some built-in robustness to rotation variance. For example, a chip rate of 4 along a 2-inch-wide thread in a 200-dpi image can be decoded with a 4-pixel offset in the corresponding 400 pixels. Rotation tolerance amounts to an offset of 1 percent of the thread length—the equivalent of only slightly more than one degree. This result would imply that a minimum of 180 threads at varying orientations would be required to completely protect an image.

To maximize resistance to rotation, several modifications to the chip signal generation were attempted. Resistance to 180-degree rotation is accomplished by forcing a rotational symmetry to the chip signal. The use of the identical chip signal for all lines

Figure 5 (A) Characterization response for varying theta values, chip length = 4, and data rate = 128; (B) characterization response before printing and after 200- and 300-dpi scans



in a thread provided for rotation tolerance of seven degrees. Unfortunately, this increased the visibility of threads to such an extent that they resembled contiguous streaks. As a compromise, each chip line was set to repeat twice, resulting in rotation tolerance of three degrees—six degrees when accounting for the 180-degree rotation (see Table 4).

Decoding. Decoding the modified DSSS signal is a challenge due to a variety of potential distortions.

Table 4 Error rates due to rotation for different chip patterns

Degrees of Rotation	Standard Chip (%)	Chip Repeated for Two Lines (%)	Chip Repeated for All Lines (%)
0.0	2.29	2.29	0.00
0.5	5.71	4.00	0.00
1.0	21.14	15.43	0.00
1.5	30.85	23.42	2.29
2.0	38.86	31.43	8.00
2.5	38.86	38.29	12.57
3.0	41.71	38.29	15.43
3.5	42.29	41.14	18.29
4.0	42.86	40.57	20.57
4.5	40.57	38.86	22.28

Table 5 Results from image resampling

Scanned dpi	Error Rate (%)
300	14.86
200	14.86
100	16.50
72	20.00
50	18.28
37	20.57
25	27.43

Table 6 Results from JPEG-encoding tests

JPEG Quality (%)	Error Rate (%)
100	14.86
75	17.71
50	15.43
25	16.00
1	19.40

However, by redundant encoding of sufficient bits, it is still possible to identify a marked image with very high probability. To this end extensive image characterizations are performed in order to find an optimal balance of encoding parameters.

Decoding requires the recovery of the phase information in a given area of the image. First, the brightness values for the encoded region of the image are multiplied by an identical chip signal, reproducing the structure of the original carrier wave with the original image data behaving as random noise. A fast Fourier transform (FFT) is applied in order to check the phase at the carrier frequency. (A more complete discussion of DSSS appears later in this paper.)

Table 7 Results from scaling tests

Scale (%)	Error Rate (%)
98.5	38.86
99.0	29.71
99.5	19.42
100.0	12.57
100.5	17.71
101.0	26.28
101.5	37.71

Decoding is performed by sampling once every chip-length by chip-length pixels. In effect, this modification creates a low-resolution encoding that can be overlaid on a high-resolution image. This decoding process is tolerant to the slight alignment errors as well as small rotational variances that are often introduced in scanning and printing. Also, it is resistant to resampling, since the encoding exists at a low enough resolution that any sampling likely to create a satisfactory reproduction of the original will also capture the encoding detail. This is essential; printed documents will be digitized with unknown dpi, orientation, etc.

Results. The encoding performed well in robustness tests. A printed version of marked currency was rescanned at varying resolutions. The scans were upsampled (interpolated between samples to generate more sample points) to 200 dpi and decoded. Table 5 shows the resulting error rates. The encoding survives even at very low resolutions: 72 dpi (screen resolution) and 50 dpi (the effective chip resolution).

Joint Photographic Experts Group (JPEG) encoding is a common lossy perceptual-compression method for image files. Tartan Threads encoding survived at JPEG quality levels varying from 100 to 1 percent. The observed error rates are shown in Table 6.

Tartan Threads does not survive geometric scaling. A change in scale of more than 1 percent pushes the error rate beyond the triggering threshold. However, this is not an issue, since it is assumed that potential counterfeiters will make copies that are identical in size to the original. Table 7 shows decoded error rates at varying scale factors.

Secure designs. A limiting factor in the current Tartan Threads implementation is that threads are being added to images that were not intended to hold them. The front of the current U.S. dollar bill, for example, has many areas with sparse printing, which limits the area available for encoding (the back of the dollar bill is ideally suited for marking). It is possible to design security documents around the fact that they must contain threads in several orientations. Although watermarking is typically approached as a process of embedding hidden information into existing documents, it is routine for organizations interested in creating secure documents (e.g., national treasuries, banks, airlines, etc.) to engineer their designs to increase security.

While the focus of this section has been on anticounterfeiting, the robust encoding necessary to survive print/scan cycles has uses in a number of applications. We have also experimented with Tartan Threads in situations where conventional bar codes are usually applied. In these applications, the advantage of applying steganographic techniques is not in increased robustness; it is aesthetic. There is no need to incorporate an unsightly bar code into the graphic design.

Tunable authentication

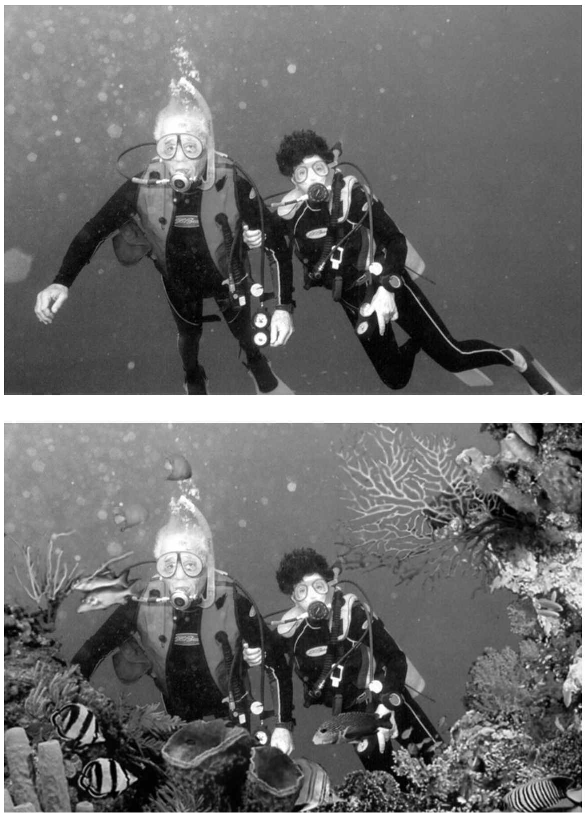
"Picture perfect" is not perfect anymore. While image manipulation is nothing new, the recent rapid dissemination of digital-image capture and manipulation technologies is creating a need to assure that the picture seen is analogous to the one that was taken, i.e., that the picture has not been tampered with. Authentication seeks to verify that the image being viewed is the same, in all important ways, to the image that was originally created for distribution. "All important ways" should be definable by the image creator or publisher.

One obvious approach to authentication is the use of a checksum. By using information-hiding techniques, a checksum can be embedded directly into an image.[30] (The use of checksums long predates modern computing. The concept comes from double-entry bookkeeping, where both rows and columns are added—these two "sums" were then compared. Allegedly, there is a form of checksums in the Qur'an[31] and they were used in tables of logarithms that were compiled in the seventeenth century.) This is challenging, since the process of embedding the checksum often changes the image statistics used to generate the checksum. While a checksum provides a robust method of determining whether or not an image has changed, this approach is limited in its applicability. Checksums afford little flexibility regarding their sensitivity—the slightest, perhaps irrelevant, change will trigger a tamper detector. Also, while a checksum can be used to determine if an image has been modified, it usually does not reveal to what extent the image has been modified. A certain amount of image manipulation, e.g., framing, tone-scale and color-balance adjustment, etc., is a routine and benign part of image reproduction. (Sometimes "routine" manipulation results in unacceptable distortions, as was the case with the O. J. Simpson "mug shot" on the cover of the June 27, 1994, issue of *Time* magazine.) An ideal authentication system would tell not only that an image has been modified, but also how and where it was modified. It should not be triggered on manipulations that have been categorized as acceptable, and detection should not require access to the original image.

Where does authentication fit into the information-hiding landscape? Authentication from the perspective of security is a problem that is beginning to interest practitioners from both academia and industry.[32,33] It is well-known that even the best information-hiding schemes are subject to degradation or removal of the embedded information if the image is subjected to certain transformations. Transformations that degrade a watermark depend on the technique used to hide the data. One can imagine placing markers in an image, or measuring image statistics, that will be modified by particular transforms. If the markers disappear, or if the statistics change noticeably, then the image must have been modified. Which markers are gone, or which statistics change, gives some insight into how the image has been manipulated. This means that watermarking schemes that do not quite resist all removal attempts can become new, useful authentication algorithms. For example, Kundur and Hatzinakos[4] have put "fragile" markers in images at various spatial-frequency bands in order to detect if there has been any change to the image at those frequencies. A variety of similar markers could be designed to detect other classes of image manipulation.

The ideal authentication system is: (1) imperceptible, (2) embedded in the image itself, (3) triggered by a selectable and arbitrary set of transforms, and (4) not incumbent on access to the original image (both to reduce bandwidth requirements and to avoid circulating the valuable original image). In keeping with this ideal, information-hiding techniques that are robust to ev-

Figure 6 An image before and after tampering



erything *except* selected transforms would be designed. This suggests a procedure where markers are placed in the image. If subsequently a marker was found distorted or missing, it would indicate that a specific disallowed transform had been performed.

Such arbitrary resistance is quite difficult to design in practice.[4,21,34,35] For example, if a marker can resist a JPEG transform, then it is likely to resist the application of a simple low-pass filter. Thus, as a first step toward building an ideal authentication system, we looked at *what* parameters in an image change when various transforms occur. Multiple characteristics of images were examined with regard to how they change under a variety of transforms. These observations led to the design of markers that resisted certain transforms.

Tunable authentication is a set of methods for detecting image tampering that localize and characterize the changes that an image has undergone. It is tunable in the sense that different image metrics can be individually tuned to trigger at different thresholds of change. This enables a detector to ignore or focus on selected classes of image manipulation. Thresholds of acceptable change can be chosen independently for each image feature. The method entails some or all of these encoding steps: (1) the original image is characterized along features such as average luminance and chrominance, standard deviation by region, spatial frequency, and entropy; (2) a database is created to store these image statistics; (3) a reference to the database is embedded in the image; and (4) additional markers are embedded in the image. The detection of tampering involves these processes: (1) the database pointer is extracted from the image; (2) the database reference is retrieved; (3) the modified image is characterized along features of interest; (4) statistics are compared with the reference statistics; and (5) markers are extracted and scrutinized.

Tunable authentication combines two information-hiding approaches to achieve a satisfactory level of both robustness and descriptiveness. Markers are embedded that are used as indicators of change, and image statistics are gathered and stored for subsequent reference.

Embedded markers. Markers are detectors that are embedded in the original image. Embedded markers are the ideal solution for tunable authentication, but difficult to design. Consequently, their use is supplemented with stored image parameters.

In our experiments, a simple modification to Patchwork was used to embed markers. The expected value, S_n, is used to assign a "confidence" as to whether or not an image has been modified and to characterize this modification along a predetermined scale. But instead of choosing a pseudorandom path, a path is chosen that is designed to distort predictably under various image modifications, such as scaling and cropping.

Several different embedded markers were implemented, each targeting a different image manipulation. Modifications to these markers are used to localize the occurrences of scaling and cropping (for implementation details, see Pogreb et al.[36]).

Parity. This marker is computed by adding up the values of the first through the seventh most-significant bits of each pixel of one or more components of the image and writing the result, modulus 2, into the least-significant bit. (An 8-bit quantization is assumed.) Pixel parity is calculated on a pixel-by-pixel basis. Plane parity is calculated by bit plane. For each bit plane, the sum of the bit values of eight pixels are written modulus 2 in successive pixels in the least-significant-bit plane. The advantage of the latter method is that it not only reveals that a change has been made, but also in what image plane the change occurred, i.e., a measure of the degree of change. The parity marker is sensitive to neither scaling nor cropping of the image, but is triggered by most other manipulations (see Figures 6 and 7).

Vertical and horizontal stripes. A marker of vertical and horizontal stripes of 0s and 1s is written into the least-significant-bit plane. This marker is used to detect scaling of the image.

Expanding pattern. A marker consisting of an expanding pattern of 0s and 1s of the form 10100100010000 . . . is written into the least-significant-bit plane. The expansion can be repeated after a short cycle or continued across the entire image. It can be oriented either horizontally or vertically or both. This marker is sensitive to scaling and also is used to detect image cropping. This pattern is generated by the following function:

$$f(n) = \operatorname{sgn}(\cos(\pi(\sqrt{1 + 8n} - 1)) - 1) + 1 \qquad (3)$$

The parity marker is not amenable to use with the Patchwork method. The expanding pattern marker is suitable for use with Patchwork. However, since the nature of the pattern reduces the data space available for placing patches, this solution is only practical for relatively large images (repeating the expanding pattern reduces the data space to only 8 percent of the original), on the order of 1000 × 1000 pixels.

Referencing image statistics. The image statistics method is dependent on data calculated from the original image. Data are sent, rather than the original image itself, since this avoids the necessity of distributing originals. These data are compared with statistics calculated directly from the suspect image. One design decision that needs to be addressed is where the data from the original should be stored. Some options are:

- *Store the data in the image itself.* Embedding data in the image itself would be ideal. However, this requires a method of storing a potentially large number of bits (several kilobytes) in the image without visibly affecting it, i.e., a solution to the core information-hiding problem. While there are numerous techniques for embedding large quantities of data in images, there is no known technique for embedding these data in a manner that is robust in light of the variety of manipulations that may occur during image manipulation. Thus we felt that, while storing data in the image is helpful for detecting a change when it occurs, we need an alternative method of storing data in order to be able to estimate the significance and the nature of the change.
- *Store the data in the form of a file header.* Another option is to attach data to the image in the form of a file header (e.g., treating the header as the cover). This method can be used to store an arbitrary quantity of data while not changing the appearance of the image. However, meta-data in a file header is fragile—it can easily be removed or replaced with meaningless data.
- *Store a pointer to a database record in the image.* Yet another option is to put data into an external database and store a pointer to the database record in the image. This option also allows an arbitrary quantity of data to be stored. It also minimizes susceptibility to attack since there are methods of securely embedding database pointers (less than 100 bits). A disadvantage of this method is that it requires access to the external database in order to utilize the data. However, given access to the Internet, this is not an insurmountable problem.

Figure 7 The results of the parity markers (triggers are shown in red)

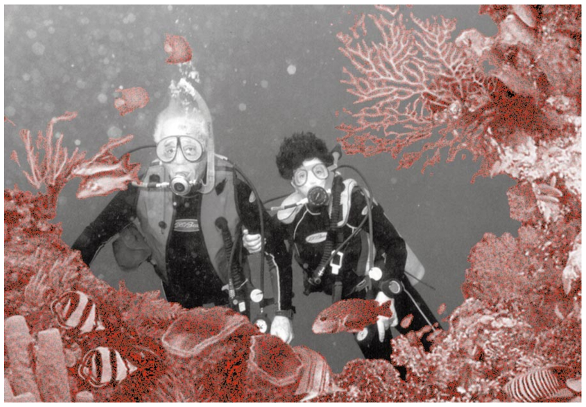


Table 8 Edge-detecting kernels

−1	2	−1	−1	−1	−1	−1	−1	2	2	−1	−1
−1	2	−1	2	2	2	−1	2	−1	−1	2	−1
−1	2	−1	−1	−1	−1	2	−1	−1	−1	−1	2

The third option, storing a pointer, was chosen due to the relatively large amount of information to be stored and the need to protect the data from tampering. The choice of an information-hiding method for storing the file pointer was Patchtrack. An image can be encoded with a desired piece of information (a reference number, URL, or an IP address and a database reference) with Patchtrack by way of pseudorandom, imperceptible alterations of intensity throughout the image. This information can then be extracted by the intended parties with a decoder. It is important to note that the encoding must be applied before the detectors are applied—otherwise, embedding the reference itself could be misconstrued as tampering.

Any number of tamper detectors can be used with the image statistics method, and since the detectors' output is stored externally, they may be used simultaneously. In our experiments, the detectors were designed based on an image decomposition into luminance and chrominance components. The detectors that we implemented include average luminance by region, standard deviation by region, histograms, edge detection by region, average Fourier transform of luminance by region, and entropy by region.

The detectors are applied to the original image and the results are stored in the database. The majority of the detectors operate on regions of the image, in which case a matrix of values is stored in the database. In implementing these "region" detectors, we divided the image into small regions (8×8 pixels corresponds to the block size used in JPEG compression).

When the "potentially tampered with" version of the image becomes available, the detectors are applied again and the results are compared to those stored in the database. For each of the regional detectors, a copy of the original image is displayed and the regions in which the value of the detector has changed significantly (where "significantly" is expressed as a tunable threshold) are highlighted.

Inferences are made about whether the image has been modified and if so, what the location and nature of the changes were likely to have been. More detailed descriptions of the individual detectors follow.

Average luminance by region. These detectors track the average luminance and chrominance of an image by region. Average luminance of a region is calculated by summing up the values of the luminance component of the image in a region, dividing by the number of pixels in the region (in our case, 64), and normalizing by the average luminance of the entire image. A normalized value is used in order to minimize the impact of global changes, such as a change in overall image brightness. These regional characteristics can be used to detect modifications such as addition, deletion, and movement of objects. A threshold can be used to localize any modified areas.

Standard deviation by region. This detector tracks the standard deviation of the image. The calculation is performed region by region. These measurements are not normalized, and thus provide slightly different information from that of the average luminance measurements. The standard deviation in a region is calculated according to the formula in Equation 4:

$$\sigma = \frac{1}{n} \sum_{i=1}^{n} (x[i] - \bar{x})^2 \qquad (4)$$

where n is the number of pixels in the regions, $x[i]$ are the individual luminance pixel values in the region, and x is the average per region. Since the detectors are calculated for each small region, they are not suitable to characterize modifications involving dimension changes. However, they react to global changes such as gamma[37] correction.

Edge detection by region. This detector tracks the edges in the image, by region. It is computed by successively convolving the luminance component of the image with the four 3×3 edge-detecting kernels (see Table 8). While these kernels are used to find horizontal, vertical, and left and right skew edges, any set of kernels can be used. The absolute value of the resulting image is then rescaled to the 0–255 range, and average value by region is calculated. The edge detector behavior is similar to that for average luminance by region except that it is not triggered by negation. This detector is not applicable to modifications that involve changes in the dimensions of the image.

Histograms. The histogram detectors track the distribution of the luminance values of the image. Our implementation calculated distributions in eight intervals roughly corresponding to the three most significant bits of each component. While the histogram detectors do not reveal anything about the location of any modification, they help to recognize that a change has occurred and to infer its nature. The histogram detectors are immutable to modifications involving a change in scale (Table 9).

Average Fourier transform of luminance by region. This detector is calculated by taking the real value FFT of the luminance component of a small region and computing the average. It behaves similarly to the average luminance by region detector in most cases. The detector is not suitable for modifications involving a change in image dimensions.

Entropy by region. This detector keeps track of the likelihood of occurrence of different values of the luminance component of a region. The entropy of a region is calculated according to the formula[38] in Equation 5:

$$H(x) = -\sum p(x) \ln[p(x)] \qquad (5)$$

This detector is not triggered either by changes in gamma or by negation. It is not suitable for modifications involving a change in the image dimensions.

Discussion. The real power of tunable authentication is in the ability to combine detectors. Table 10 aggregates all of our results. From this table it is possible to see patterns of redundancy and significance from which inferences can be made about the character of any image modifications. Interesting and useful combinations abound, e.g., a modification that triggers histogram but not entropy detection would indicate a change to gamma, and a modification that triggers standard deviation but not histogram detection indicates scaling. Consideration of the embedded markers adds to the ability to characterize changes. For example, there is no apparent difference between the triggering due to adding objects and image cropping. The parity marker can be used to distinguish between these modifications. Either the Fourier transform detector or the expanding pattern marker can be used to characterize the direction and magnitude of image scaling. Reading the columns in Table 10 as a binary code, each of the modifications can be associated with a unique bit pattern. For example, adding objects results in a pattern of 1 1 1 1 1 1 1 1, while moving an object results in a pattern of 1 1 1 1 1 1 0 1. The statistical redundancy of the various methods leads to further characterization of any modifications.

Table 9 The results of the histogram detector. Note that scaling does not trigger the detector.

Bin	Original	Modified	Scaled Nearest Neighbor	Scaled Bicubic
0–30	26.50	10.81	26.50	26.44
31–60	22.21	23.82	22.21	22.23
61–90	9.65	21.97	9.64	9.72
91–120	6.18	9.70	6.18	6.20
121–150	8.08	10.20	8.08	8.04
151–180	10.36	10.02	10.35	10.34
181–210	10.71	6.91	10.71	10.68
211–255	6.31	6.58	6.31	6.35

Tunable authentication provides a characterization of the modifications to which an image may have been subjected (and in many cases this characterization is localized to a particular region in the image). There are many image characterizations that may be more useful or efficient than the ones just described. Also, there is an opportunity to explore in more detail how markers change as a result of image modifications and how to detect these changes. Nonetheless, the work described is sufficiently mature to be applied to many current problems of image tampering.

Tunable authentication might find broader application in areas outside the domain of verification and authentication. There are many production and publishing applications that could benefit from a system of image characterization that is independent of any single vendor's solution. Here the goal is not to increase security, but to facilitate auditing and process control.

Midlevel vision techniques

As described in the previous section, techniques for authenticating images seek to exploit naturally occurring "markers" to generate unique, descriptive keys. These keys should be easy to recover and should be robust in the presence of image manipulations common to production environments (e.g., cropping, changes in color composition, gamma and tone-scale correction). These techniques rely on manipulation of low-level image statistics built upon the raw pixel

Table 10 Aggregate results (“triggered” may be false trigger)

	Average	Standard Deviation	Histogram	Edge	FFT	Entropy	Parity	Expand
Adding Objects	triggered	triggered	triggered	triggered	triggered	triggered	triggered	triggered
Moving Objects	triggered	triggered	triggered	triggered	triggered	triggered	none	triggered
Adjusting Gamma	slight	triggered	triggered	none	triggered	none	triggered	none
Desaturating	triggered	triggered	triggered	none	none	none	none	none
Cropping	triggered	triggered	triggered	triggered	triggered	triggered	none	triggered
Scaling	triggered	triggered	none	triggered	triggered	triggered	triggered	triggered
Blurring	triggered	triggered	slight	triggered	triggered	triggered	triggered	none
Rotating	triggered	triggered	none	triggered	triggered	triggered	none	triggered
Displacing	triggered	triggered	triggered	triggered	triggered	triggered	none	triggered
Mirroring	triggered	triggered	none	triggered	triggered	triggered	none	triggered
Negating	triggered	triggered	triggered	none	triggered	none	triggered	none

data (e.g., local entropy, histograms, etc.). These statistics are not reflective of how groups of pixels are perceived as objects—they are limited in both their resilience and descriptive capacity.

Machine-vision techniques for midlevel visual analysis can produce image representations that are more descriptive and are robust against distortion, are image specific, can be compactly parameterized, and can be altered with minimum perceptible effect on the image. These properties can be exploited to support the recovery of inherent identification (ID) markers useful for authentication.

While it seems natural to focus attention on high-level vision, machine emulation of human visual performance is not yet up to the task. Machine-vision systems cannot recognize most objects in the sense that humans do. And for cases when they can (e.g., faces), the computational load is prohibitive, the recognition too fragile, and the number of recognizable objects is too small.

As an alternative, we turn to midlevel vision where the “objects” are simpler constructs such as layers, oriented surfaces, and regions grouped in accordance with some metric of coherence. In all cases, these midlevel objects are “blobs” of arbitrary shape. While not objects in the human sense, the constructs of midlevel vision are regarded as intermediary symbols. On the one hand, they are built on top of primitive transformations performed on the rasterized photosensor data. On the other hand, they serve as building blocks for high-level symbols that humans commonly regard as objects.

We next examine methods for using these constructs as a vehicle to robustly identify images. Details for a representative set of midlevel vision routines for use in the subsequent analysis are presented. How these routines are applied to read a “fingerprint” is described.

Analysis using midlevel vision. Image analysis using midlevel vision routines differs from low-level image analysis in a number of important ways—one important difference being granularity. In low-level vision, transform values are computed on a uniform pixel raster. In midlevel vision, the transforms are computed over an arbitrarily shaped region. As an example, while computation of a pixel-by-pixel motion-vector field would be considered a low-level routine, grouping of those vectors to identify a planar patch undergoing affine motion would be considered a midlevel routine. In low-level routines, sensor noise can cause substantial fluctuation of the output. However, the blobs produced by midlevel routines can be increasingly robust against distortion in the underlying pixel values.

Common examples of midlevel visual analysis include techniques for segmentation based on coherence of motion, texture, or normalized color. Other examples are routines that group pixels based on depth, lighting, or observed surface patterns. In all cases, the output of a midlevel routine is a blob—a data structure consisting of: (1) a description of the region (often an *alpha* mask); and (2) a functional description of the activity within this region. (In this context, our use of the word “blob” derives from its more common use in the computational vision community—the tokenized bases of the 2½-dimensional sketch.[39]

Midlevel visual routines are increasingly popular in systems for scene analysis and for image coding. In data hiding the advantages of midlevel techniques are that the blobs can often be compactly described by a few parameters, that these parameters are recoverable, and that they are somewhat robust to changes, such as scaling and rotation, in the underlying pixel statistics.

For this work, an analysis technique is chosen that characterizes the difference between an image and a predefined model employing a vocabulary of five distinct types of blobs: (1) coherent motion, (2) occlusion, (3) lighting, (4) multiplicative gain, and (5) additive bias. This technique was originally demonstrated in the context of an object-based image coding application.[40] There, the goal was to deconstruct an image into a constituent set of visual events, represent these events as blobs, and compactly code these blobs separately for transmission. The current application also begins with a predefined model and current image. The iterative analysis routine successively approximates the image as a series of transformations performed on the model. Each transformation is expressed as a blob, with each iteration producing only one blob. The analysis is adaptive in that there are no biases on the type or position of a blob for any given iteration.

Table 11 lists the blob types and the corresponding formula for the functional approximations. For the lighting blob, the change in lighting is modeled as a multiplicative gain approximated by a 2nd order polynomial—one for each of the three color channels. For the motion blob, the displacement is modeled as an affine warp of a planar surface, and the estimate is a six-parameter affine description. In the occlusion blob, the new pixels are JPEG encoded. In the gain and bias blobs, the change is modeled (respectively) as a uniform multiplicative gain and additive bias.

Figure 8 illustrates the analysis procedure applied to real images. Starting with an image and the predefined model, the first iteration recovers the global background motion (in this case, a translation of 18 pixels horizontally and two pixels vertically). In the second iteration, the shadow is approximated as an arbitrary region over which a single scalar gain is applied. Later, in the 17th iteration, much of the person is approximated as an occluding object.

Computing the characteristic vector. The previous example suggests how midlevel vision analysis can be applied to characterization. Various properties of the blobs, such as the position, shape, and param-

Figure 8 Selected bids from a coding iteration

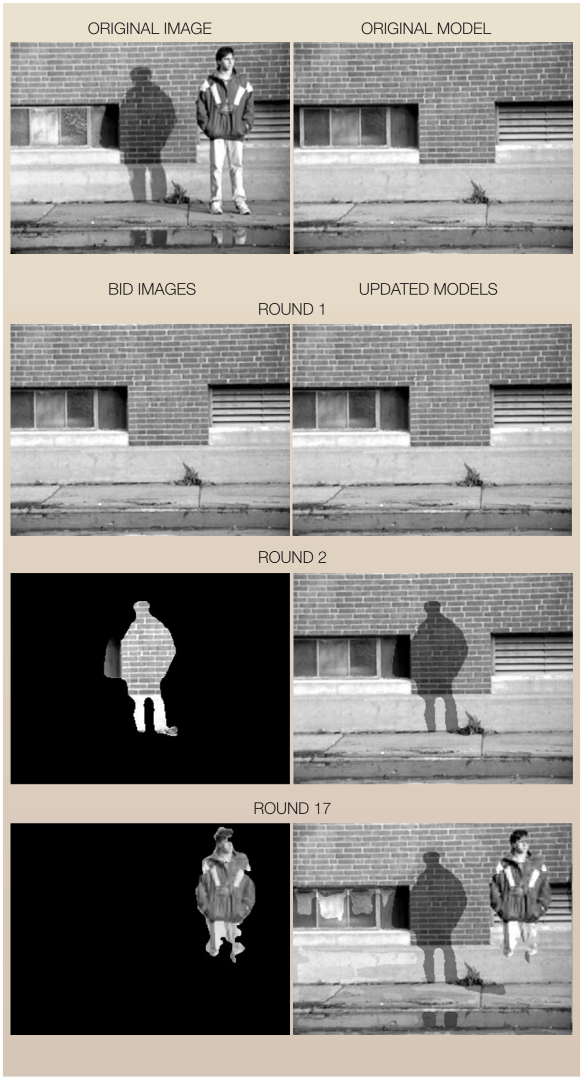
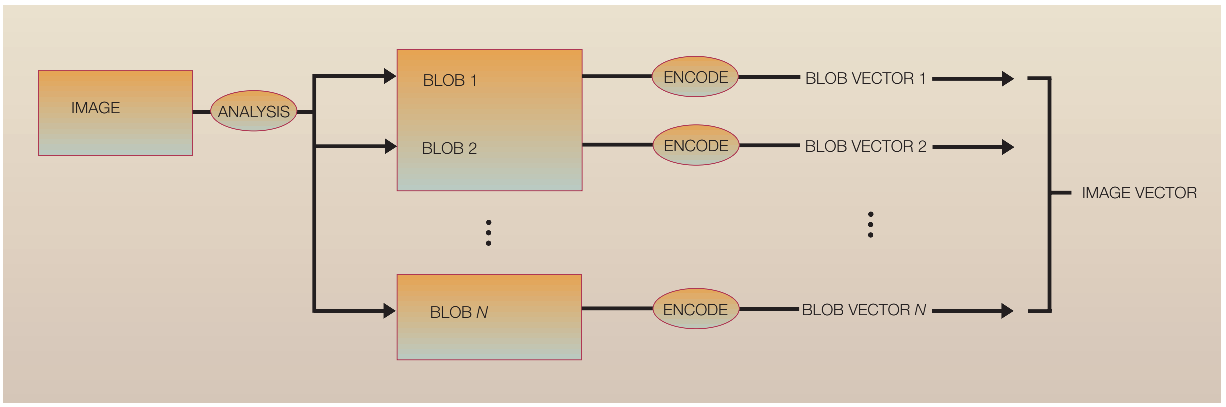


Table 11 Models of blob behavior

Blob Type	Functional Model
Lighting	$R_{gain} = R_0 + \alpha_0 x + \alpha_1 y$
	$G_{gain} = G_0 + \alpha_0 x + \alpha_1 y$
	$B_{gain} = B_0 + \alpha_0 x + \alpha_1 y$
Motion	$V_x = A_0 + A_x x + A_y y$
	$V_y = B_0 + B_x x + B_y y$
Occlusion	All pixels JPEG encoded
Gain	$R_{gain}, G_{gain}, B_{gain}$
Bias	$R_{bias}, G_{bias}, B_{bias}$

Figure 9 Computing the characteristic vector for authentication. Midlevel image analysis produces a set of *N* blobs. A separate blob vector is computed for each blob and the ensemble is assembled to form the image vector.



eters of the functional models, can be assembled into vectors that are unique to the image. Ideally, a characteristic vector would be: (1) easy to compute, (2) of reasonable size, (3) unique to the image, yet (4) robust against changes that a human would regard as immaterial.

These last two properties suggest that the vector should reflect those properties of the image that the human viewer regards as both salient and immutable—a requirement that naturally favors statistics constructed on higher-image abstractions. The incorporation of midlevel vision is a step in this direction.

Additionally, the particular midlevel analysis technique employed here has a number of advantages:

- The analysis technique is a component of a larger analysis-synthesis system. The analysis of the image into blobs is therefore reversible in that the image can be resynthesized from the preexisting model and the blobs. This is useful in applications where the blobs are themselves altered as a vehicle for embedding data.
- The blobs are produced sequentially and for any given image-model pair, that sequence should be unique.
- The difference between the image and the model can be substantial. Changes in lighting, camera position, or occluding objects can all be estimated with varying degrees of fidelity.

Figure 9 illustrates how blobs from midlevel vision can be used to construct a characteristic vector for an image. The image analysis routine produces a series of arbitrarily shaped blobs. Blob-specific statistics are built. The data from several blobs are aggregated into a single vector. This process is repeated at the receiver where the recovered vector is compared with the precomputed one.

For a given blob, a variety of statistics can be computed from the shape of the blob and functional descriptions. The attributes available from the functional estimate depend on the blob type. Here, the tokens are derived by permuting combinations of the parameters shown in Table 11. For example, statistics for motion blobs can be derived from the six parameters of the affine warp model. Useful attributes of the shape include the size, the centroid, the moments, the central moments, and the principal component axes. Similarly, distinctive features of the bounding contour include points of extremal curvature and the intervening degrees of convexity and concavity.

Simple concatenation of all available blob statistics would produce ID vectors of impractical length. Further, any ID vector that is truly characteristic of the image should incorporate information from several blobs. In this work, the characteristic ID is defined as a *macro vector*, consisting of a separate vector from each of five blobs. Each blob vector is itself a mixture of statistics from the shape and the functional approximations.

The canonical organization of a blob vector is shown in Figure 10. The blob number refers to the iteration of the analysis that produced the blob. The blob type indicates from which of the five possible blob types the current blob is drawn. The region statistics describe the size and the two-dimensional centroid of the bitmap that defines the shape of the blob. The contour statistics are prepared by first subdividing the contour into a series of segments bounded by points of extremal curvature. For each segment, the relative size of the segment is computed, as well as a flag indicating the convexity or concavity of the contour segment. Finally, a maximum of six 16-bit words are reserved for statistics built on the functional description of the blob.

Statistics built on the functional description. The number of tokens that can be built on the functional model of the blob depends upon the blob type. Recall that each blob has an associated functional model (Table 11). Entries in the characteristic vector are built from the parameters of these models as shown in Table 12.

For lighting blobs, five parameters are extracted—one multiplicative constant per color channel (red, green, blue) plus two weights describing the degree of spatial variation. Data for the motion blobs are drawn from the six parameters describing the affine motion model. For the occlusion blobs, the mean value is computed for each of the three color channels. The gain and bias blobs also report a single value of the three color components.

Statistics built on the region mask. Similarly, the region statistics are computed from the blob's bitmap using the simple formulas in Equations 6, 7, and 8:

$$\text{relative size of region} = \frac{1}{N}\sum_{i}^{N} b(i) \quad (6)$$

$$\text{normalized } x \text{ centroid} = \frac{1}{width \times S}\sum_{i}^{N} x(i) \times b(i) \quad (7)$$

$$\text{normalized } y \text{ centroid} = \frac{1}{height \times S}\sum_{i}^{N} y(i) \times b(i) \quad (8)$$

Figure 10 Organization of blob vector. Descriptive vector for a blob contains statistics built on shape and on functional model. Shape descriptor is subdivided into tokens based on region and tokens built on the contour.

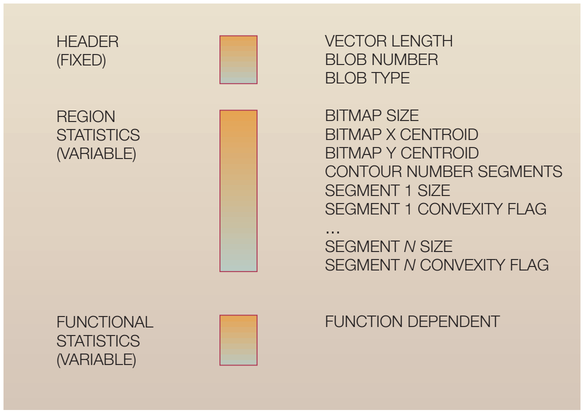


Table 12 Parameters extracted from the blob's function module

Blob Type	Functional Model
Lighting	$R_0, G_0, B_0, \alpha_0, \alpha_1$
Motion	A_0, A_x, A_y, B_x, B_y
Occlusion	$R_{mean}, G_{mean}, B_{mean}$
Gain	R_0, G_0, B_0
Bias	R_0, G_0, B_0

where $b(i)$ equals 1 for a foreground pixel and 0 for a background pixel, N is the number of pixels, S is $\sum_i^N b(i)$, $x(i)$ is the x coordinate for pixel i, and $y(i)$ is the y coordinate for pixel i.

Statistics built on the contour. Building the tokens from the contours is more involved. The basic strategy is to qualitatively characterize a contour by locating the dominant points of maximal curvature, dividing the contour into segments bounded by these dominant points, and then measuring the degree of convexity or concavity of these segments. The shortcoming of this approach is the sensitivity to changes in scale, orientation, and local distortion. To compensate for this, we employ the scale-space filtering technique advanced by Pei and Lin[41] to locate those dominant points that are stable with respect to scale.

Figure 11 illustrates the representation of the contour for each of four levels of detail. The points of

Figure 11 Scale-space contour representation. Samples of a scale-space representation of the contour outlining the six New England states. Four levels of detail are selected by controlling σ in Gaussian filter: (A) σ=7; (B) σ=20; (C) σ=38; (D) σ=77. Dominant points are marked by circles along the contour.

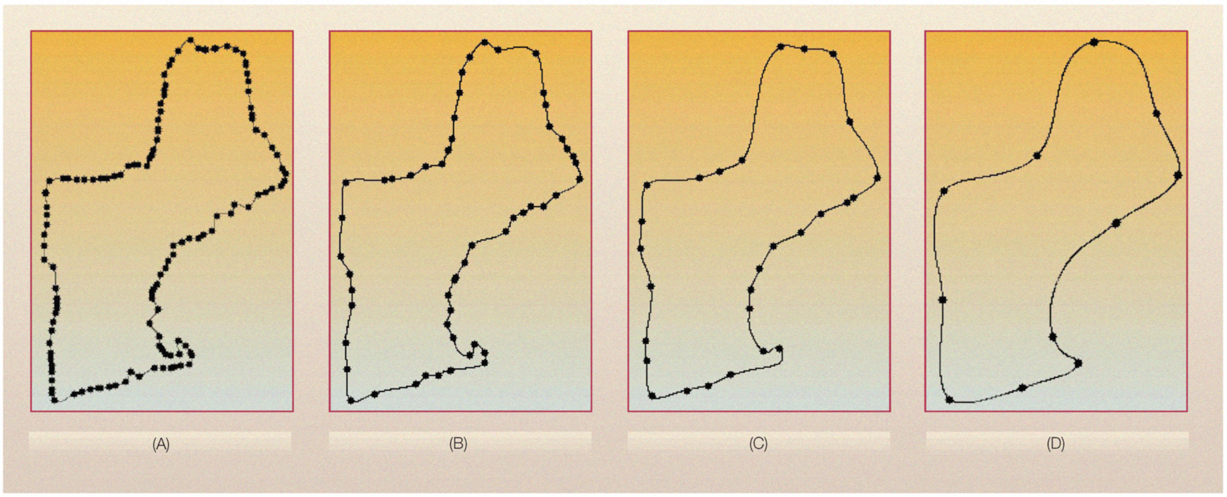
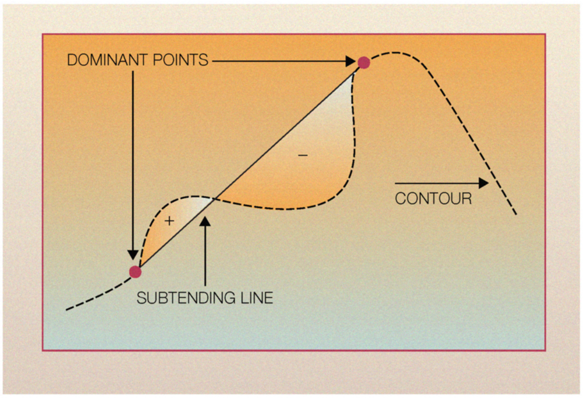


Figure 12 Measuring convexity/concavity. For each contour segment defined by a pair of dominant points, the convexity or concavity is measured as the integral along the subtended contour segment. The straight-line connection between the end points delineates the polarity of the area.



extremal curvature are superimposed on the contours. As the user traverses the scale space from fine to coarse detail, the number of dominant points decreases. This decrease is variable and often; a fixed number of dominant points are constant across a substantial range of scale. For this work, a description is sought that is stable over the largest range of scale, with the fewest possible dominant points in a constellation. For instances in which these two requirements conflict, the selection is made based on a cost function that weights the two conditions.

Once the dominant points have been located, the convexity or concavity of the resulting contour segments remains to be computed. Here, this is approximated by simply drawing a straight line between the endpoints of each segment and integrating along the subtended contour segment as shown in Figure 12.

Comparing vectors. Use of the characteristic vector for authentication requires defining a distance metric for use in comparing one vector to another. Obviously, identical images will produce identical vectors. However, the methods for computing a vector are numerically sensitive enough that modest differences between images will produce modest differences between their respective vectors. A distance metric has to be defined such that visual differences that a human perceives as unimportant result in numerically small values of measured distance.

With the image vector defined as macro vector of five blob vectors, the task of comparing two image vectors naturally divides into two subtasks: (1) measuring the differences between individual blob vectors; and (2) combining these differences to produce an aggregate difference between the two image vectors.

Measuring distance between blobs. The difference between two blob vectors is computed as the itemized difference between the subcomponents, i.e., as the weighted difference between the functional models, the region tokens, and the contour tokens. Therefore, only blobs of identical type can be compared. For example, a motion blob can only be compared against another motion blob; it cannot be compared against a lighting blob.

The distance between the functional descriptions is computed as the L2 norm[42] of the distance between each of the components. For the case of two lighting blobs, the functional distance would be computed using Equation 9.

$$\sqrt{(\Delta A)^2 + (\Delta B)^2 + (\Delta C)^2 + (\Delta D)^2 + (\Delta E)^2 + (\Delta F)^2} \tag{9}$$

Similarly, the difference between the shape portions of two vectors is computed as the L2 norm of the difference between each of the component parts. Equations 10, 11, 12, and 13 illustrate this breakdown. Given two contours to be compared, the first step is to apply match filtering to establish a correspondence between the contour segments. Once the correspondence is defined, the differences between the segments are measured and accumulated.

$$\Delta shape = \sqrt{\Delta region^2 + \Delta contour^2} \tag{10}$$

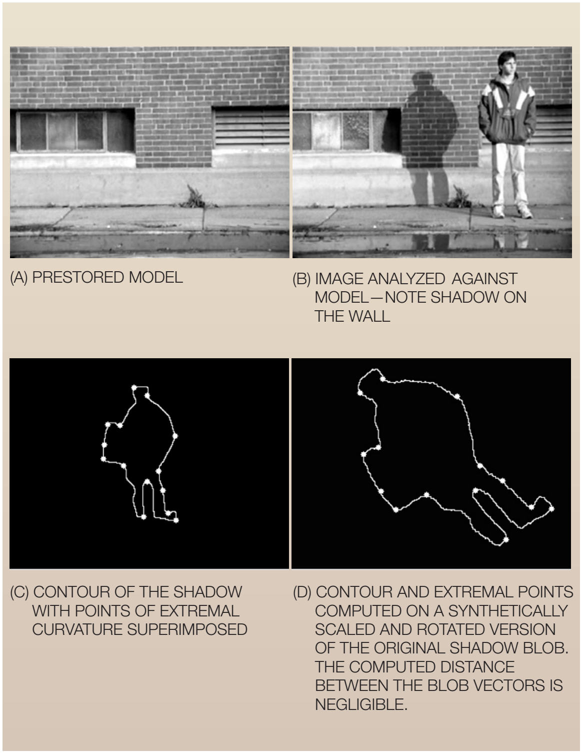
$$\Delta region = \sqrt{\Delta size^2 + \Delta xcentroid^2 + \Delta ycentroid^2} \tag{11}$$

$$\Delta contour = \sqrt{\Delta seg_1^2 + \Delta seg_2^2 + \ldots + \Delta seg_N^2} \tag{12}$$

$$\Delta seg = \sqrt{\Delta length^2 + \Delta concavity^2} \tag{13}$$

This distance metric with a contour represented as extremal points of curvature yields a measure that is surprisingly robust in light of typical image manipulations. Consider the illustration in Figure 13: 13A and 13B show the model and the image. One of the blobs returned by the analysis system is a gain blob, which models the shadow. Figure 13C shows the contour of the shadow blob upon which the detected points of extremal curvature are superimposed. Figure 13D shows the recomputed contour after the blob has been synthetically scaled and rotated about its centroid. The measured distance between the two blob vectors is vanishingly small.

Figure 13 Robustness of blob representation to rotation


(A) PRESTORED MODEL
(B) IMAGE ANALYZED AGAINST MODEL—NOTE SHADOW ON THE WALL
(C) CONTOUR OF THE SHADOW WITH POINTS OF EXTREMAL CURVATURE SUPERIMPOSED
(D) CONTOUR AND EXTREMAL POINTS COMPUTED ON A SYNTHETICALLY SCALED AND ROTATED VERSION OF THE ORIGINAL SHADOW BLOB. THE COMPUTED DISTANCE BETWEEN THE BLOB VECTORS IS NEGLIGIBLE.

Measuring distance between image vectors. As in the case of tunable authentication, authentication of an image involves comparing two characteristic image vectors; the one that has been precomputed (the reference image) and the one computed "on the fly" (the modified image). A "best match" strategy is employed to do this comparison. For a given blob vector in the reference image vector, the closest match is found among the blob vectors of the modified image vector. The differences between blob vectors are aggregated to arrive at the total difference between the two image vectors, with unmatched blob vectors contributing a predefined maximum.

The application of midlevel vision to information hiding is at an immature state. There is further work to be done in image characterization, blob characterization, blob comparison, etc. Future work includes the manipulation of blob vectors in order to embed information in images.

Conclusion

Looking forward, there are two questions regarding information hiding: (1) What is its potential as a technology? and (2) What is its potential as an application solution?

The technology at the heart of most information-hiding technologies, spread spectrum, was invented by actress Hedy Lamarr and composer George Antheil[26,43] in 1942. While its use as the basis of information-hiding techniques has led to numerous refinements and improvements in efficiency and robustness, it is a well-worn technology. In contrast, object- or semantic-based techniques are relatively new. (Steganographic techniques have tended to lag somewhat behind those of the field of image compression. Techniques based upon transform coding have only been in vogue in steganography since the late 1990s, whereas they have been incorporated into compression standards [e.g., JPEG] since the late 1980s.) As object-based compression protocols, such as MPEG4,[44] become more commonplace, it is likely that object-based techniques will be embraced by the information-hiding community. Time will tell whether these approaches will be as fruitful when applied to the information-hiding problem space.

Most of the applications of information hiding are likewise well worn. The classic application is secret communication. This is the application that motivated Lamarr and Antheil as it has motivated others throughout history. (The first recorded use of steganography was by the Spartans, who used a cipher device as early as 400 BC.[45]) The "growth" application of information hiding in the 1990s was watermarking. Watermarks are known to have existed in Italy before the end of the thirteenth century.[46,47] More recently they have been considered as a panacea for intellectual-property protection on the Internet (however, there is some skepticism about the ultimate utility of information hiding thus applied[48]).

We have explored diverse applications of information hiding that can be characterized by exchanging robustness with bandwidth, including marking playing cards,[20] monitoring newspaper reading habits, tracking images in a production process,[36] adding audio links to photographs, providing a channel for device-to-device communication,[8] embedding decoding information, interaction methods, or multimedia object behavior,[47] and a variety of augmentations to presentation.[9,10,49,50] These embedded data examples are of the class of *signals with a sense of themselves*. We expect these will be the growth applications of information hiding over the next ten years, especially in light of network computing.

Acknowledgment

This work was sponsored in part by the MIT Media Laboratory's News in the Future research consortium and IBM Corporation. The authors would also like to thank the anonymous reviewers for their helpful comments.

Cited references and notes

1. *Information Hiding: First International Workshop*, R. J. Anderson, Editor, Lecture Notes in Computer Science **1174**, Isaac Newton Institute, Cambridge, England, Springer-Verlag (May 1996).
2. W. Bender, D. Gruhl, N. Morimoto, and A. Lu, "Techniques for Data Hiding," *IBM Systems Journal* **35**, Nos. 3&4, 313–336 (1996).
3. R. J. Anderson and J.-H. Lee, "Jikzi: A New Framework for Secure Publishing," *Proceedings of the Security Protocols Workshop*, Cambridge, UK (April 1999).
4. D. Kundur and D. Hatzinakos, "Digital Watermarking for Telltale Tamper-Proofing and Authentication," *Proceedings of the IEEE Special Issue on Identification and Protection of Multimedia Information* **87**, No. 7, 1167–1180 (July 1999).
5. M. P. Quelez, "Content-Based Integrity Protection of Digital Images," *Security and Watermarking of Multimedia Contents*, P. W. Wong and E. J. Delp, Editors, The Society for Imaging Science and Technology (IS&T) and the International Society for Optical Engineering (SPIE) **3657**, San Jose, CA (January 25–27, 1999), pp. 85–93.
6. T. Demuth and A. Rieke, "Securing the Anonymity of Content Providers in the World Wide Web," *Security and Watermarking of Multimedia Contents*, P. W. Wong and E. J. Delp, Editors, The Society for Imaging Science and Technology (IS&T) and the International Society for Optical Engineering (SPIE) **3657**, San Jose, CA (January 25–27, 1999), pp. 494–502.
7. R. J. Anderson, R. M. Needham, and A. Shamir, "The Steganographic File System," *Information Hiding: Second International Workshop*, D. Aucsmith, Editor, Lecture Notes in Computer Science **1525**, Springer-Verlag, Portland, OR (April 15–17, 1998), pp. 73–82.
8. V. Gerasimov and W. Bender, "Things That Talk: Using Sound for Device-to-Device and Device-to-Human Communication," *IBM Systems Journal* **39**, Nos. 3&4, 530–546 (2000, this issue).
9. E. H. Adelson, *Digital Signal Encoding and Decoding Apparatus*, U.S. Patent No. 4,939,515 (July 3, 1990).
10. W. F. Schreiber, A. B. Lippman, E. H. Adelson, and A. N. Netravali, *Receiver-Compatible Enhanced Definition Television System*, U.S. Patent No. 5,010,405 (April 23, 1991).
11. R. J. Anderson and F. A. P. Petitcolas, *Information Hiding & Digital Watermarking: An Annotated Bibliography*, see http://www.cl.cam.ac.uk/~fapp2/steganography/bibliography/.
12. F. Hurtung and M. Kutter, "Multimedia Watermarking Techniques," *Proceedings of IEEE Special Issue on Identification and Protection of Multimedia Information* **87**, No. 7, 1069–1107 (July 1999).
13. DataHiding home page, http://www.trl.ibm.co.jp/projects/s7730/Hiding/.
14. Signal Processing Lab of the Swiss Federal Institute of Technology (EPFL/LTS), École Polytechnique Fédérale de Lausanne, see http://ltssg3.epfl.ch/publications/.
15. University of Geneva, Switzerland, Computer Science De-

partment, Computer Vision Group (CUI), see http://cui.unige.ch/~vision/Publications/.
16. A. Piva, University of Florence, http://cosimo.die.unifi.it/~piva/Watermarking/watermark.html.
17. F. Mintzer and G. W. Braudaway, "If One Watermark Is Good, Are More Better?" *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, IEEE Signal Processing Society, Phoenix, AZ (March 15–19, 1999), pp. 2067–2070.
18. J. Fridrich and M. Goljan, "Comparing Robustness of Watermarking Techniques," *Security and Watermarking of Multimedia Contents*, P. W. Wong and E. J. Delp, Editors, The Society for Imaging Science and Technology (IS&T) and the International Society for Optical Engineering (SPIE) **3657**, San Jose, CA (January 25–27, 1999), pp. 214–225.
19. W. Bender and D. Gruhl, "Information Hiding to Foil the Casual Counterfeiter," *Information Hiding: Second International Workshop*, D. Aucsmith, Editor, Lecture Notes in Computer Science **1525**, Springer-Verlag, Portland, OR (April 15–17, 1998), pp. 1–15.
20. R. Hwang, *A Robust Algorithm for Information Hiding in Digital Pictures*, M.Eng. thesis, MIT, Cambridge, MA (May 1999).
21. S. Pereira and T. Pun, "Fast Robust Template Matching for Affine Resistant Image Watermarking," *International Workshop on Information Hiding*, Lecture Notes in Computer Science **1768**, Springer-Verlag, Dresden, Germany (September 29–October 1, 1999), pp. 200–210.
22. F. Paiz, *Tartan Threads: A Method for the Real-Time Digital Recognition of Secure Documents in Ink Jet Printers*, M.Eng. thesis, MIT, Cambridge, MA (May 1999).
23. L. M. Marvel, C. G. Boncelet, and C. T. Retter, "Reliable Blind Information Hiding for Images," *Information Hiding: Second International Workshop*, D. Aucsmith, Editor, Lecture Notes in Computer Science **1525**, Springer-Verlag, Portland, OR (April 15–17, 1998), pp. 48–62.
24. A. Herrigel, J. J. K. Ó Ruanaidh, H. Petersen, S. Pereira, and T. Pun, "Secure Copyright Protection Techniques for Digital Images," *Information Hiding: Second International Workshop*, D. Aucsmith, Editor, Lecture Notes in Computer Science **1525**, Springer-Verlag, Portland, OR (April 15–17, 1998), pp. 169–190.
25. J. Fridrich, "Robust Digital Watermarking Based on Key-Dependent Basis Functions," *Information Hiding: Second International Workshop*, D. Aucsmith, Editor, Lecture Notes in Computer Science **1525**, Springer-Verlag, Portland, OR (April 15–17, 1998), pp. 143–157.
26. M. K. Simon, J. K. Omura, R. A. Scholtz, and B. K. Levitt, *Spread Spectrum Communications Handbook*, McGraw-Hill, New York (1994).
27. A. Drake, *Fundamentals of Applied Probability Theory*, McGraw-Hill, New York (1967).
28. A. N. Netravali and B. G. Haskell, *Digital Pictures: Representation, Compression, and Standards*, Applications of Communications Theory, Plenum Publishers, New York (1995).
29. W. Bender, "Adaptive Color Coding Based on Spatial/Temporal Features," *Proceedings of SPIE—the International Society of Optical Engineering* **901**, Los Angeles, CA (January 13–15, 1988), pp. 253–257.
30. S. Walton, "Image Authentication for a Slippery New Age," *Dr. Dobb's Journal*, 18–26 (April 1995).
31. Personal communications with Ken Haase and Alan Wexelblat (January 2000).
32. C.-Y. Lin, Bibliography of Multimedia Authentication Research Papers, ADVENT Lab, Columbia University, see http://www.ctr.columbia.edu/~cylin/auth/bibauth.html.
33. J. Dittmann, "Overview," *Multimedia and Security Workshop at ACM Multimedia*, Orlando, FL (October 30–31, 1999), see http://www.darmstadt.gmd.de/mobile/acm99/.
34. F. A. P. Petitcolas, "Attacks on Copyright Marking Systems," *Information Hiding: Second International Workshop*, D. Aucsmith, Editor, Lecture Notes in Computer Science **1525**, Springer-Verlag, Portland, OR (April 15–17, 1998), pp. 219–239.
35. F. A. P. Petitcolas and R. J. Anderson, "Evaluation of Copyright Marking Systems," *Proceedings IEEE Multimedia Computing and Systems* **1**, Florence, Italy (June 7–11, 1999), pp. 574–579.
36. S. Pogreb, W. Bender, and D. Gruhl, "Tunable Tamper Proofing," unpublished technical report, MIT Media Laboratory, see http://nif.www.media.mit.edu/DataHiding/ttp.pdf.
37. *Gamma* represents a numerical parameter that describes the nonlinearity of intensity reproduction. See Charles Poynton's Gamma FAQ: http://www.inforamp.net/~poynton/GammaFAQ.html.
38. See http://mathworld.wolfram.com/Entropy.html/.
39. D. Marr, *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*, W. H. Freeman and Company, New York (1983).
40. V. M. Bove and W. Butera, "The Coding Ecology: Image Coding via Competition Among Experts," *Proceedings of the Picture Coding Symposium Among Experts* (*PCS'99*), Corvalis, OR (April 21–23, 1999), pp. 403–406.
41. S.-C. Pei and C.-N. Lin, "The Detection of Dominant Points on Digital Curves by Scale Space Filtering," *Pattern Recognition* **25**, No. 11, 1307–1344 (1992).
42. The L2 norm is also known as the Euclidian norm. See http://mathworld.wolfram.com/L2-Norm.html.
43. H. K. Markey and G. Antheil, *Secret Communication System*, U.S. Patent No. 2,292,387 (August 11, 1942).
44. *The MPEG4 Standard*, International Organization for Standardization ISO/IEC JTC1/SC29/WG11 Coding of Moving Pictures and Audio ISO/IEC JTC1/SC29/WG11 N3342 (March 2000).
45. D. Kahn, *The Codebreakers—The Story of Secret Writing*, Scribner, New York (1996).
46. "Watermarks," *Encyclopædia Britannica*, http://www.britannica.com.
47. T. Leary, "Cryptology in the 15th and 16th Century," *Cryptologia* **XX**, No. 3, 223–242 (July 1996).
48. R. J. Anderson and F. A. P. Petitcolas, "On the Limits of Steganography," *IEEE Journal of Selected Areas in Communications* **16**, No. 4, 474–481 (May 1998).
49. N. Abramson and W. Bender, "Context-Sensitive Multimedia," *Proceedings of the International Society for Optical Engineering (SPIE)* **1785**, Washington, DC (September 10–11, 1992), pp. 122–32.
50. W. Bender and P. Chesnais, "Network Plus," *Proceedings of SPIE—the International Society for Optical Engineering* **900**, Los Angeles, CA (January 12–13, 1988), pp. 81–86.

Accepted for publication June 26, 2000.

Walter Bender *MIT Media Laboratory, 20 Ames Street, Cambridge, Massachusetts 02139-4307 (electronic mail: walter@media.mit.edu).* Mr. Bender is a senior scientist at the MIT Media Laboratory and principal investigator of the laboratory's News in the Future consortium. He received the B.A. degree from Harvard University in 1977 and joined the Architecture Machine Group at MIT in 1978. He received the M.S. degree from MIT in 1980. Mr. Bender is a founding member of the Media Laboratory.

William Butera *MIT Media Laboratory, 20 Ames Street, Cambridge, Massachusetts 02139-4307 (electronic mail: bill@media.mit.edu).* A native of Washington, DC, Mr. Butera received his B.S. and M.S. degrees from MIT in 1982 and 1988. In 1982 he joined the Research and Development Department of ITT Industries in Stuttgart, Germany, where he worked on video coding schemes for broadband ISDN (Integrated Services Digital Network). In September of 1986 he joined the Movies program at MIT's Media Lab as a research assistant and basketball coach. From 1988 through 1994, he was a system designer in the Concept Engineering Department at Intermetall in Freiburg, Germany, where he developed digital video components for the consumer electronics market. In 1995, he joined MIT's Media Lab, where he works as a research assistant on programming models and algorithms for dense, decentralized computing ensembles. An early participant in MPEG, he has authored several articles and holds five patents in the field of digital formats for video compression and storage. His interests include architectures for parallel processing, image coding, and machine vision.

Daniel Gruhl *IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, California 95120-6099 (electronic mail: dgruhl@almaden.ibm.com).* Dr. Gruhl is a research staff member at the IBM Almaden Research Center, where he is a member of the Exploratory Database Systems group. He received his doctorate degree in electrical engineering from MIT in 2000; his research was done at the MIT Media Laboratory.

Raymond Hwang *Harvard Medical School, 25 Shattuck Street, Boston, Massachusetts 02115 (electronic mail: raymond_hwang@student.hms.harvard.edu).* Born and raised in Cleveland, Ohio, Mr. Hwang received his B.S. and M.Eng. degrees in computer science and electrical engineering from MIT. Currently he is a student at Harvard Medical School.

Fernando J. Paiz *There, 165 Jefferson Drive, Menlo Park, California 94025 (electronic mail: fpaiz@there.com).* Mr. Paiz graduated from MIT in June 1999, simultaneously earning bachelor's and master's degrees in computer science and electrical engineering and a bachelor's degree in theater arts. Since graduation he has been applying his multimedia interests and skills toward his role as a software engineer at There, a Silicon Valley start-up company.

Sofya Pogreb *MIT Media Laboratory, 20 Ames Street, Cambridge, Massachusetts 02139-4307 (electronic mail: spogreb@mit.edu).* Ms. Pogreb graduated from MIT in June, 2000, with degrees in both computer science and management. She began working for McKinsey & Company in Palo Alto, California, in the summer of this year. Past work experience has included internships at Hewlett-Packard Company and Intel Corporation, as well as research projects at the MIT Media Lab and the Lab for Computer Science.