Image Processing

Rama Chellappa
Azriel Rosenfeld
University of Maryland

GLOSSARY

Compression Reduction of the amount of data used to represent an image, by compact encoding or by approximation.
Description Information about image parts and their properties and relations.
Digitization Conversion of an image into a discrete array of numbers for computer processing.
Enhancement Processing of an image to improve its appearance.
Matching Comparison of images for purposes of pattern recognition, registration, stereopsis, or motion analysis.
Recognition Recognition of objects in an image by comparing image descriptions with object models.
Reconstruction Computation of cross sections of an image or volume, given a set of its projections.
Recovery Estimation of the orientation of a surface from the shading, texture, or shape of the corresponding region of an image.
Representation Mathematical description of image data.
Restoration Processing of an image to correct for the effects of degradations.
Segmentation Partitioning of an image into homogeneous regions; extraction of features such as edges or curves from an image.

COMPUTERS are used to process images for many purposes. Image processing is the computer manipulation of images to produce more useful images; subfields of image processing include image compression or coding, image enhancement and restoration, and image reconstruction from projections; examples of applications include compression of DVD movies, restoration of Hubble telescope images, and medical imaging. The goal of image analysis is to produce a description of the scene that gave rise to the

image; examples of applications are reading documents (character recognition), counting blood cells on a microscope slide, detecting tumors in chest X rays, producing land use maps from satellite images, detecting defects in printed circuits, and guiding robots in navigating or in manipulating objects. In addition to processing images in the visible spectrum, acquisition, processing and analysis of infrared, synthetic aperture radar, medical, hyperspectral, and range images have become important over the last 20 years due to the emergence of new sensors and applications.

I. INTRODUCTION

This chapter deals with the manipulation and analysis of images by computer. In image processing, both the input and the output are images, the output being, for example, an approximated or improved version of the input. In image analysis (also known by such names as pictorial pattern recognition, image understanding, and computer vision), the input is an image and the output is (typically) a description of the scene that gave rise to the image. Computer graphics, which is not covered in this chapter, is the inverse of image analysis: The input is a scene description, and the output is an image of the scene as it would appear from a given viewpoint.

An image is defined by specifying how its value (brightness, color, etc.) varies from point to point—in other words, by a function of two variables defined over an "image plane." Before an image can be processed and analyzed by (digital) computer, it must be converted into a discrete array of numbers each of which represents the value at a given point. This process of conversion is called digitization (Section II).

A digitized image can be viewed as a matrix of gray-level values. To understand/analyze the structure of this matrix, image models and image transforms have been used. Image models attempt to describe the image data quantatively, while image transforms enable the analysis of the image data in the transform domain for various applications such as compression, restoration, and filtering.Image models and representations are discussed in Section III.

To represent the input image with sufficient accuracy, the array of numbers must usually be quite large—for example, about 500×500 in the case of a television image. Image compression (or coding) deals with methods of reducing this large quantity of data without sacrificing important information about the image (Section IV).

One of the central goals of image processing is to improve the appearance of the image—for example, by increasing contrast, reducing blur, or removing noise. Image enhancement (Section V) deals with methods of improving the appearance of an image. More specifically, image restoration (Section VI) is concerned with estimating image degradations and attempting to correct them.

Another important branch of image processing is image reconstruction from projections (Section VII). Here we are given a set of images (e.g., X rays) representing projections of a given volume, and the task is to compute and display images representing cross sections of that volume.

Comparison or matching of images is an important tool in both image processing and analysis. Section VIII discusses image matching and registration and depth measurement by comparison of images taken from different positions (stereomapping).

Section IX summarizes methods for the analysis of image sequences. Techniques for motion compensation, detection and tracking of moving objects, and recovery of scene structure from motion using optic flow and discrete features are discussed.

The brightness of an image at a point depends on many factors, including the illumination, reflectivity, and surface orientation of the corresponding surface point in the scene. Section X discusses methods of recovering these "intrinsic" scene characteristics from an image by analyzing shading, texture, or shapes in the image.

Image analysis usually begins with feature detection or segmentation—the extraction of parts of an image, such as edges, curves, or regions, that are relevant to its description. Techniques for singling out significant parts of an image are reviewed in Section XI. Methods of compactly representing image parts for computer manipulation, as well as methods of decomposing image parts based on geometric criteria and of computing geometric properties of image parts, are treated in Section XII.

Section XIII deals with image description, with an emphasis on the problem of recognizing objects in an image. It reviews properties and relations, relational structures, models, and knowledge-based image analysis.

A chapter such as this would not be complete without some discussion of architectures designed for efficient processing of images. The eighties witnessed an explosion of parallel algorithms and architectures for image processing and analysis; especially noteworthy were hypercube-connected machines. In the early nineties attention was focused on special processors such as pyramid machines. Recently, emphasis is being given to embedded processors and field-programmable gate arrays. Section XIV presents a summary of these developments.

The treatment in this chapter is concept-oriented; applications are not discussed, and the use of mathematics has been minimized, although some understanding of Fourier transforms, stochastic processes, estimation theory, and linear algebra is occasionally assumed. The Bibliography

lists basic textbooks, as well as a recent survey article that celebrated 50 years of progress in image processing and analysis.

Since an earlier version of this chapter appeared in 1986, image processing and analysis have grown so dramatically in depth and breadth that it is nearly impossible to cover all their subareas in detail. To adhere to our page limitations, we have made a conscientious selection of topics for discussion. We avoid discussion of topics such as scanners, display devices, and hardware and software issues. On the other hand, since the mideighties, much to our delight, the field has been supported by a strong underlying analytical framework based on mathematics, statistics, and physics. We point out the influence of these fields on image processing, analysis, understanding, and computer vision.

II. DIGITIZATION

Digitization is the process of converting an image into a discrete array of numbers. The array is called a digital image, its elements are called pixels (short for "picture elements"), and their values are called gray levels. (In a digital color image, each pixel has a set of values representing color components.)

Digitization involves two processes: sampling the image value at a discrete grid of points and quantizing the value at each of these points to make it one of a discrete set of gray levels. In this section we briefly discuss sampling and quantization.

A. Sampling

In general, any process of converting a picture into a discrete set of numbers can be regarded as "sampling," but we assume here that the numbers represent the image values at a grid of points (or, more precisely, average values over small neighborhoods of these points). Traditionally, rectangular uniform sampling lattices have been used, but depending on the shape of the image spectrum, hexagonal, circular, or nonuniform sampling lattices are sometimes more efficient.

The grid spacing must be fine enough to capture all the detail of interest in the image; if it is too coarse, information may be lost or misrepresented. According to the sampling theorem, if the grid spacing is d, we can exactly reconstruct all periodic (sinusoidal) components of the image that have period $2d$ or greater (or, equivalently, "spatial frequency" $1/2d$ or fewer cycles per unit length). However, if patterns having periods smaller than $2d$ are present, the sampled image may appear to contain spurious patterns having longer periods; this phenomenon is called aliasing. Moiré patterns are an everyday example of aliasing, usually arising when a scene containing short-period patterns is viewed through a grating or mesh (which acts as a sampling grid).

Recent developments in the design of digital cameras enable us to acquire digital images instantly. In addition, many image processing operations such as contrast enhancement and dynamic range improvement are being transferred to the acquisition stage.

B. Quantization

Let z be the value of a given image sample, representing the brightness of the scene at a given point. Since the values lie in a bounded range, and values that differ by sufficiently little are indistinguishable to the eye, it suffices to use a discrete set of values. It is standard practice to use 256 values and represent each value by an 8-bit integer $(0, 1, \ldots, 255)$. Using too few discrete values give rise to "false contours," which are especially conspicuous in regions where the gray level varies slowly. The range of values used is called the grayscale.

Let the discrete values ("quantization levels") be $z_1, \ldots, z_k$. To quantize z, we replace it by the z_j that lies closest to it (resolving ties arbitrarily). The absolute difference $|z - z_j|$ is called the quantization error at z.

Ordinarily, the z's are taken to be equally spaced over the range of possible brightnesses. However, if the brightnesses in the scene do not occur equally often, we can reduce the average quantization error by spacing the z's unequally. In fact, in heavily populated parts of the brightness range, we should space the z's closely together, since this will result in small quantization errors. As a result, in sparsely populated parts of the range, the z's will have to be farther apart, resulting in larger quantization errors; but only a few pixels will have these large errors, whereas most pixels will have small errors, so that the average error will be small. This technique is sometimes called tapered quantization.

Quantization can be optimized for certain distributions of brightness levels, such as Gaussian and double-exponential. Another class of techniques, known as moment-preserving quantizers, does not make specific assumptions about distributions. Instead, the quantizers are designed so that low-order moments of the brightness before and after quantization are required to be equal. These scalar quantizers do not exploit the correlation between adjacent pixels. Vector quantization (VQ), in which a small array of pixels is represented by one of several standard patterns, has become a popular method of image compression on its own merits, as well as in combination with techniques such as subband or wavelet decomposition. A major difficulty with VQ techniques is their computational complexity.

III. REPRESENTATION

Two dominant representations for images are two-dimensional (2-D) discrete transforms and various types of image models. In the former category, linear, orthogonal, separable transforms have been popular due to their energy-preserving nature and computational simplicity. When images are represented using discrete transforms, the coefficients of the expansion can be used for synthesis, compression, and classification. The two most often used discrete transforms are the discrete Fourier transform (due to its FFT implementation) and discrete cosine transform (due to its FFT-based implementation and its adoption in the JPEG image compression standard). If minimizing the mean squared error between the original image and its expansion using a linear orthogonal transform is the goal, it can be shown that the Karhunen–Loeve transform (KLT) is the optimal transform, but the KLT is not practical, since it requires eigencomputations of large matrices. Under a circulant covariance structure, the DFT is identical to the KLT.

All the 2-D discrete transforms mentioned above analyze the data at one scale or resolution only. Over the last 20 years, new transforms that split the image into multiple frequency bands have given rise to schemes that analyze images at multiple scales. These transforms, known as wavelet transforms, have long been known to mathematicians. The implementation of wavelet transforms using filter banks has enabled the development of many new transforms.

Since the early eighties, significant progress has been made in developing stochastic models for images. The most important reason is the abstraction that such models provide of the large amounts of data contained in the images. Using analytical representations for images, one can develop systematic algorithms for accomplishing a particular image-related task. As an example, model-based optimal estimation-theoretic principles can be applied to find edges in textured images or remove blur and noise from degraded images. Another advantage of using image models is that one can develop techniques to validate a given model for a given image. On the basis of such a validation, the performance of algorithms can be compared.

Most statistical models for image processing and analysis treat images as 2-D data, i.e., no attempt is made to relate the 3-D world to its 2-D projection on the image. There exists a class of models, known as image-formation models, which explicitly relate the 3-D information to the 2-D brightness array through a nonlinear reflectance map by making appropriate assumptions about the surface being imaged. Such models have been the basis for computer graphics applications, can be customized for particular sensors, and have been used for inferring shape from shading and other related applications. More recently, accurate prediction of object and clutter models (trees, foliage, urban scenes) has been recognized as a key component in model-based object recognition as well as change detection. Another class of models known as fractals, originally proposed by Mandelbrot, is useful for representing images of natural scenes such as mountains and terrain. Fractal models have been successfully applied in the areas of image synthesis, compression, and analysis.

One of the earliest model-based approaches was the multivariate Gaussian model used in restoration. Regression models in the form of facets have been used for deriving hypothesis tests for edge detection as well as deriving the probability of detection. Deterministic 2-D sinusoidal models and polynomial models for object recognition have also been effective.

Given that the pixels in a local neighborhood are correlated, researchers have proposed 2-D extensions of time-series models for images in particular, and for spatial data in general, since the early 1950s. Generalizations have included 2-D causal, nonsymmetric half-plane (NSHP), and noncausal models.

One of the desirable features in modeling is the ability to model nonstationary images. A two-stage approach, regarding the given image as composed of stationary patches, has attempted to deal with nonstationarities in the image. A significant contribution to modeling nonstationary images used the concept of a dual lattice process, in which the intensity array is modeled as a multilevel Markov random field (MRF) and the discontinuities are modeled as line processes at the dual lattice sites interposed between the regular lattice sites. This work has led to several novel image processing and analysis algorithms.

Over the last 5 years, image modeling has taken on a more physics-based flavor. This has become necessary because of sensors such as infrared, laser radar, SAR, and foliage-penetrating SAR becoming more common in applications such as target recognition and image exploitation. Signatures predicted using electromagnetic scattering theory are used for model-based recognition and elimination of changes in the image due to factors such as weather.

IV. COMPRESSION

The aim of image compression (or image coding) is to reduce the amount of information needed to specify an image, or an acceptable approximation to an image. Compression makes it possible to store images using less memory or transmit them in less time (or at a lower bandwidth).

A. Exact Encoding

Because images are not totally random, it is often possible to encode them so that the coded image is more compact than the original, while still permitting exact reconstruction of the original. Such encoding methods are generally known as "lossless coding."

As a simple illustration of this idea, suppose that the image has 256 gray levels but they do not occur equally often. Rather than representing each gray level by an 8-bit number, we can use short "codes" for the frequently occurring levels and long codes for the rare levels. Evidently this implies that the average code length is relatively short, since the frequent levels outnumber the rare ones. If the frequent levels are sufficiently frequent, this can result in an average code length of less than eight bits. A general method of constructing codes of this type is called Shannon–Fano–Huffman coding.

As another example, suppose that the image has only a few gray levels, say two (i.e., it is a black-and-white, or binary, image: two levels might be sufficient for digitizing documents, e.g., if we can distinguish ink from paper reliably enough). Suppose, further, that the patterns of black and white in the image are relatively simple—for example, that it consists of black blobs on a white background. Let us divide the image into small blocks, say 3×3. Theoretically, there are $2^9 = 512$ such blocks (each of the nine pixels in a block can have gray level either 0 or 1), but not all of these combinations occur equally often. For example, combinations like

$$\begin{matrix}1&1&1\\1&1&1\\1&1&1\end{matrix} \quad \text{or} \quad \begin{matrix}0&1&1\\0&1&1\\0&0&1\end{matrix} \quad \text{or} \quad \begin{matrix}1&1&0\\0&0&0\\0&0&0\end{matrix} \quad \text{or} \quad \begin{matrix}0&0&0\\0&0&0\\0&0&0\end{matrix}$$

should occur frequently, but combinations like

$$\begin{matrix}1&0&1\\0&1&0\\1&0&0\end{matrix}$$

should occur rarely or not at all. By using short codes to represent the frequent blocks and longer codes for the rare ones, we can reduce (to much less than 9) the average number of bits required to encode a block. This approach is sometimes called area character coding or block coding. Other methods of compactly encoding such images, including run length coding and contour coding, are discussed in Section 12 when we consider methods of compactly representing image parts.

B. Approximation

The degree of compression obtainable by exact encoding is usually small, perhaps of the order of 2:1. Much higher degrees of compression, of 20:1 or greater, can be obtained if we are willing to accept a close approximation to the image (which can, in fact, be virtually indistinguishable from the original).

The fineness of sampling and quantization ordinarily used to represent images are designed to handle worst cases. Adequate approximations can often be obtained using a coarser sampling grid or fewer quantization levels. In particular, sampling can be coarse in image regions where the gray level varies slowly, and quantization can be coarse in regions where the gray level varies rapidly. This idea can be applied to image compression by using "adaptive" sampling and quantization whose fineness varies from place to place in the image. Alternatively, we can use Fourier analysis to break the image up into low-frequency and high-frequency components. The low-frequency component can be sampled coarsely, the high-frequency component can be quantized coarsely, and the two can then be recombined to obtain a good approximation to the image. Still another idea is to add a pseudorandom "noise" pattern, of amplitude about one quantization step, to the image before quantizing it and then subtract the same pattern from the image before displaying it; the resulting "dithering" of the gray levels tends to break up false contours, so that quantization to fewer levels yields an acceptable image.

C. Difference Coding and Transform Coding

In this section we discuss methods of transforming an image so as to take greater advantage of both the exact encoding and approximation approaches.

Suppose we scan the image (say) row by row and use the gray levels of a set of preceding pixels to predict the gray level of the current pixel—by linear extrapolation, for example. Let $\hat{z}_i$ be the predicted gray level of the ith pixel and z_i its actual gray level. If we are given the actual gray levels of the first few pixels (so we can initiate the prediction process) and the differences $z_i - \hat{z}_i$ between the actual and the predicted values for the rest of the pixels, we can reconstruct the image exactly. The differences, of course, are not a compact encoding of the image; in fact, they can be as large as the largest gray level and can be either positive or negative, so that an extra bit (a sign bit) is needed to represent them. However, the differences do provide a basis for compact encoding, for two reasons.

1. The differences occur very unequally; small differences are much more common than large ones, since large, unpredictable jumps in gray level are rare in most types of images. Thus exact encoding

techniques, as described in Section IV.A, can be used to greater advantage if we apply them to the differences rather than to the original gray levels.
2. When large differences do occur, the gray level is fluctuating rapidly. Thus large differences can be quantized coarsely, so that fewer quantization levels are required to cover the range of differences.

The main disadvantage of this difference coding approach is the relatively low degree of compression that it typically achieves.

More generally, suppose that we apply an invertible transformation to an image—for example, we take its discrete Fourier transform. We can then encode or approximate the transformed image, and when we want to reconstruct the original image, we apply the inverse transformation to the approximated transform. Evidently, the usefulness of this transform coding approach depends on the transformed image being highly "compressible."

If we use the Fourier transform, it turns out that for most classes of images, the magnitudes of the low-frequency Fourier coefficients are very high, whereas those of the high-frequency coefficients are low. Thus we can use a different quantization scheme for each coefficient—fine quantization at low frequencies, coarse quantization at high frequencies. (For sufficiently high frequencies, the magnitudes are so small that they can be discarded.) In fact, the quantization at high frequencies can be very coarse because they represent parts of the image where the gray level is fluctuating rapidly. When the image is reconstructed using the inverse transform, errors in the Fourier coefficients are distributed over the entire image and so tend to be less conspicuous (unless they are very large, in which case they show up as periodic patterns). An example of transform coding using the discrete cosine transform is shown in Fig. 1.

Difference and transform coding techniques can be combined, leading to hybrid methods. For example, one can use a 1-D transform within each row of the image, as well as differencing to predict the transform coefficients for each successive row by extrapolating from the coefficients for the preceding row(s). Difference coding is also very useful in the compression of time sequences of images (e.g., sequences of television frames), since it can be used to predict each successive image by extrapolating from the preceding image(s).

D. Recent Trends

The difference and transform compression schemes described above attempt to decorrelate the image data at a single resolution only. With the advent of multiresolution representations (pyramids, wavelet transforms, subband decompositions, and quadtrees), hierarchical compression schemes have become the dominant approach. Subband and wavelet compression schemes provide better distributions of bits across the frequency bands than are possible in single-resolution schemes. Also, much higher compression factors with improved peak to signal-to-noise ratios are being achieved using these schemes. Additional features such as progressive compression schemes also make these methods more attractive. An example of subband coding is also shown in Fig. 1.

During the early years of image compression research, although compressed images were transmitted over noisy channels, source and channel coding aspects were treated independently. Due to the emergence of wired and wireless networks, the problems of compression and transmission are now solved using a joint source/channel framework, with the goal of obtaining the best possible compression results for the given channel. Depending on whether the state of the channel is known a priori or is estimated on-line, different strategies are possible. These developments, in addition to reenergizing image compression research, have also led to new coding techniques such as turbo coding.

Compression of image sequences has been investigated for more than 20 years for applications ranging from video telephony to DVD movies. One of the ways to achieve the very high compression factors required in these applications is to exploit temporal redundancy, in addition to the spatial redundancy that is present in still images. Temporal redundancy is removed by using pixel- or object-based motion compensation schemes, skipping frames, or conditional replenishment. During the early years, pel-recursive displacement estimates were used to assign motion vectors to blocks of pixels. Recently, object-based schemes are being introduced. An example of moving object detection in a video frame is shown in Fig. 2.

One of the clear signs that technology has contributed to a commercial product is that standards exist for its implementation. Image compression, because of its usefulness in a large number of applications, is an area that has seen standardization of its technology. For still image compression, standards such as those developed by the Joint Photographic Experts Group (JPEG) and the Joint Bi-level Photographic Experts Group (JBIG) are in daily use. A new standard (JPEG-2000), which includes wavelet transform compression, will be available soon. For compressing image sequences, many standards are available, including (but not limited to) H. 261, H. 263, MPEG-1, MPEG-2, and MPEG-4, and new standards such as MPEG-7 are being developed. These new and improved standards for still and sequence compression reflect the widespread acceptance of research results and technological advances obtained in universities, research laboratories, and industry.

(a)

FIGURE 1 Image compression. (a) Original image; (b) compressed image, using the discrete cosine transform; (c) compressed image, using a subband coding scheme.

V. ENHANCEMENT

The general goal of image enhancement is to improve the appearance of an image so that it can be easier to use. The appropriate method of enhancement depends on the application, but certain general-purpose methods are applicable in many situations. In this section we describe some basic methods of grayscale modification (or contrast stretching), blur reduction, shading reduction, and noise cleaning.

A. Grayscale Modification

When the illumination of a scene is too high or too low or the objects in a scene have reflectivities close to that of the background, the gray levels in the resulting image will occupy only a portion of the grayscale. One can improve the appearance of such an image by spreading its gray levels apart to occupy a wider range. This process of contrast stretching does not introduce any new information, but it may make fine detail or low-contrast objects more clearly

(b)

FIGURE 1 *(continued)*

visible, since spreading apart indistinguishable (e.g., adjacent) gray levels makes them distinguishable.

Even if an image occupies the entire grayscale, one can spread apart the gray levels in one part of the grayscale at the cost of packing them closer together (i.e., combining adjacent levels) in other parts. This is advantageous if the information of interest is represented primarily by gray levels in the stretched range.

If some gray levels occur more frequently than others (e.g., the gray levels at the ends of a grayscale are usually relatively uncommon), one can improve the overall contrast of the image by spreading apart the frequently occurring gray levels while packing the rarer ones more closely together; note that this stretches the contrast for most of the image. (Compare the concept of tapered quantization in Section II.B.) The same effect is achieved by requantizing the image so that each gray level occurs approximately equally often; this breaks up each frequently occuring gray level into several levels, while compressing sets of consecutive rarely occuring levels into a single level. Given an

(c)

FIGURE 1 *(continued)*

image, we can plot a graph showing how often each gray level occurs in the image; this graph is called the image's histogram. The method of requantization just described is called histogram flattening. It is sometimes preferable to requantize an image so as to give it a histogram of some other standard shape.

Since humans can distinguish many more colors than shades of gray, another useful contrast stretching technique is to map the gray levels into colors; this technique is called pseudo-color enhancement.

B. Blur Reduction

When an image is blurred, the ideal gray level of each pixel is replaced by a weighted average of the gray levels in a neighborhood of that pixel. The effects of blurring can be reversed, to a first approximation, by subtracting from the gray level of each pixel in the blurred image a weighted average of the gray levels of the neighboring pixels. This method of "sharpening" or deblurring an image is sometimes referred to as Laplacian processing, because the

(a)

(b)

FIGURE 2 Moving object detection in a video frame. (a) Original frame; (b) detected object.

difference between a pixel's gray level and the average of its neighbors' gray levels is a digital approximation to the Laplacian (sum of second partial derivatives) of the image.

Blurring weakens the high-frequency Fourier components of an image more than it does the low ones. Hence high-emphasis frequency filtering, in which the high-frequency components are strengthened relative to the low ones, has a deblurring effect. Such filtering will also enhance noise, since noise tends to have relativity strong high-frequency components; one should avoid strengthening frequencies at which the noise is stronger than the information-bearing image detail.

C. Shading Reduction

The illumination across a scene usually varies slowly, while the reflectivity may vary rapidly from point to point. Thus illumination variations, or shading, give rise primarily to low-frequency Fourier components in an image, while reflectivity variations also give rise to high-frequency components. Thus one might attempt to reduce shading effects in an image by high-emphasis frequency filtering, weakening the low-frequency components in the image's Fourier transform relative to the high-frequency components. Unfortunately, this simple approach does not work, because the illumination and reflectivity information is combined multiplicatively rather than additively; the brightness of the scene (and hence of the image) at a point is the product of illumination and reflectivity, not their sum. Better results can be obtained using a technique called homomorphic filtering. If we take the logarithm of the gray level at each point of the image, the result is the sum of the logarithms of the illumination and reflectivity; in other words, the multiplicative combination has been transformed into an additive one. We can thus take the Fourier transform of the log-scaled image and apply high-emphasis filtering to it. Taking the inverse Fourier transform and the antilog at each point then gives us an enhanced image in which the effects of shading have been reduced.

D. Noise Cleaning

Noise that is distinguishable from the image detail is relatively easy to remove from an image. For example, if the image is composed of large objects and the noise consists of high-contrast specks ("salt and pepper"), we can detect the specks as pixels that are very different in gray level from (nearly) all of their neighbors and remove them by replacing each such pixel by the average of its neighbors. As another example, if the noise is a periodic pattern, in the Fourier transform of the image it gives rise to a small set of isolated high values (i.e., specks); these can be detected and removed as just described, and the inverse Fourier transform can be applied to reconstruct an image from which the periodic pattern has been deleted. (This process is sometimes called notch filtering.)

Image noise can also be reduced by averaging operations. If we have several copies of an image that are identical except for the noise (e.g., several photographs or television frames of the same scene), averaging the copies reduces the amplitude of the noise while preserving the image detail. In a single image, local averaging (of each pixel with its neighbors) will reduce noise in "flat" regions of the image, but it will also blur the edges or boundaries of regions. To avoid blurring, one can first attempt to detect edges and then average each pixel that lies near an edge only with those of its neighbors that lie on the same side of the edge. One need not actually decide whether an edge is present but can simply average each pixel only with those of its neighbors whose gray levels are closest to its own, since these neighbors are likely to lie on the same side of the edge (if any) as the pixel. [Better results are obtained by selecting, from each symmetric pair of neighbors, the one whose gray level is closer to that of the pixel. If the image's histogram has distinctive peaks (see Section XI.A), averaging each pixel with those of its neighbors whose gray levels belong to the same peak also has a strong smoothing effect.] Another approach is to examine a set of wedge-shaped neighborhoods extending out from the pixel in different directions and to use the average of that neighborhood whose gray levels are least variable, since such a neighborhood is likely to lie on one side of an edge. A more general class of approaches uses local surface fitting (to the gray levels of the neighbors) rather than local averaging; this too requires modifications to avoid blurring edges.

Another class of noise-cleaning methods is based on rank ordering, rather than averaging, the gray levels of the neighbors of each pixel. The following are two examples of this approach.

1. *Min–max filtering.* Suppose that the noise consists of small specks that are lighter than their surroundings. If we replace each pixel's gray level by the minimum of its neighbors' gray levels, these specks disappear, but light objects also shrink in size. We now replace each pixel by the maximum of its neighbors; this reexpands the light objects, but the specks do not reappear. Specks that are darker than their surroundings can be removed by an analogous process of taking a local maximum followed by a local minimum.
2. *Median filtering.* The median gray level in a neighborhood is the level such that half the pixels in the neighborhood are lighter than it and half are darker. In a "flat" region of an image, the median is usually close to the mean; thus replacing each pixel by the median of its neighbors has much the same effect as local averaging. For a pixel near a (relatively straight) edge, on the other hand, the median of the neighbors will usually be one of the neighbors on the same side of the edge as the pixel, since these neighbors are in the majority; hence replacing the pixel by the median of the neighbors will not blur the edge. Note that both min–max and median filtering destroy thin features such as lines, curves, or sharp corners; if they are used on an image that contains such features, one should first attempt to detect them so they can be preserved.

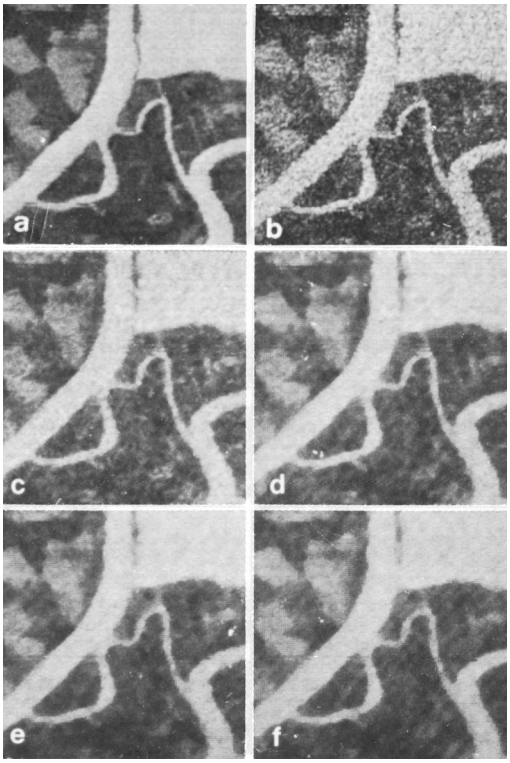


FIGURE 3 Image smoothing by iterated median filtering. [Figure 7 in the previous edition.]

Median filters belong to a class of filters known as order-statistic or rank order-based filters. These filters can effectively handle contamination due to heavy-tailed, non-Gaussian noise, while at the same time not blurring edges or other sharp features, as linear filters almost always do. An example of iterated median filtering is shown in Fig. 3.

VI. RESTORATION

The goal of image restoration is to undo the effects of given or estimated image degradations. In this section we describe some basic methods of restoration, including photometric correction, geometric correction, deconvolution, and estimation (of the image gray levels in the presence of noise).

A. Photometric Correction

Ideally, the gray levels in a digital image should represent the brightnesses at the corresponding points of the scene in a consistent way. In practice, however, the mapping from brightness to gray level may vary from point to point—for example, because the response of the sensor is not uniform or because the sensor collects more light from scene points in the center of its field of view than from points in the periphery ("vignetting").

We can estimate the nonuniformity of the sensor by using images of known test objects. A nonnoisy image of a uniformly bright surface should have a constant gray level; thus variations in its gray level must be due to sensor nonuniformity. We can measure the variation at each point and use it to compute a correction for the gray level of an arbitrary image at that point. For example, if we regard the nonuniformity as an attenuation by the factor $a(x, y)$ at each point, we can compensate for it by multiplying the gray level at (x, y) by $1/a(x, y)$.

B. Geometric Correction

The image obtained by a sensor may be geometrically distorted, by optical aberrations, for example. We can estimate the distortion by using images of known test objects such as regular grids. We can then compute a geometric transformation that will correct the distortion in any image.

A geometric transformation is defined by a pair of functions $x' = \phi(x, y)$, $y' = \psi(x, y)$ that map the old coordinates (x, y) into new ones (x', y'). When we apply such a transformation to a digital image, the input points (x, y) are regularly spaced sample points, say with integer coordinates, but the output points (x', y') can be arbitrary points of the plane, depending on the nature of the transformation. To obtain a digital image as output, we can map the output points into the nearest integer-coordinate points. Unfortunately, this mapping is not one-to-one; some integer-coordinate points in the output image may have no input points mapped into them, whereas others may have more than one.

To circumvent this problem, we use the inverse transformation $x = \phi(x', y')$, $y = \psi(x', y')$ to map each integer-coordinate point of the output image back into the input image plane. This point is then assigned a gray level derived from the levels of the nearby input image points—for example, the gray level of the nearest point or a weighted average of the gray levels of the surrounding points.

C. Deconvolution

Suppose that an image has been blurred by a known process of local weighted averaging. Mathematically, such a blurring process is described by the convolution $(g = h * f)$ of the image f with the pattern of weights h. (The value of $h * f$ for a given shift of h relative to f is obtained by pointwise multiplying them and summing the results.)

By the convolution theorem for Fourier transforms, we have $G = HF$, where F, G, H are the Fourier transforms

of f, g, h, respectively. Thus in principle we can restore the unblurred f by computing $F = G/H$ and inverse Fourier transforming. (At frequencies where H has zeros, G/H is undefined, but we can take $F = G = 0$.) This process is called inverse filtering. If h is not given, we can estimate it from the blurred images of known test objects such as points, lines, or step edges.

The simple deconvolution process just described ignores the effects of noise. A more realistic model for a degraded image is $g = h * f + n$, where n represents the noise. If we try to apply inverse filtering in this situation we have $G = HF + N$, so that $G/H = F + N/H$. Thus at high spatial frequencies, where the noise is stronger than the information-bearing image detail, the results of inverse filtering will be dominated by the noise. To avoid this, the division G/H should be performed only at relatively low spatial frequencies, whereas at higher frequencies G should be left intact.

A more general process known as least-squares filtering or Wiener filtering can be used when noise is present, provided the statistical properties of the noise are known. In this approach, g is deblurred by convolving it with a filter m, chosen to minimize the expected squared difference between f and $m * g$. It can be shown that the Fourier transform M of m is of the form $(1/H)[1/(1 + S)]$, where S is related to the spectral density of the noise; note that in the absence of noise this reduces to the inverse filter: $M = 1/H$. A number of other restoration criteria lead to similar filter designs.

Other methods of deblurring can be defined that assume some knowledge about the statistical properties of the noise. A typical approach is to find an estimate $\hat{f}$ of f such that the "residual" $g - h * \hat{f}$ has the same statistical properties as n. This problem usually has many solutions, and we can impose additional constraints on $\hat{f}$ (that it be nonnegative, "smooth," etc.). Deblurring techniques have also been developed in which the choice of filter dependson the local characteristics of the image or in which the blur itself is assumed to vary in different parts of the image.

D. Estimation

Suppose that an image has been corrupted by the addition of noise, $g = f + n$, where the statistics of the noise are known. We want to find the optimum estimate $\hat{f}$ of the image gray level at each point. If the image has no structure, our only basis for this estimate is the observed gray level g at the given point; but for real-world images, neighboring points are not independent of one another, so that the estimates at neighboring points can also be used in computing the estimate at a given point.

One approach to image estimation, known as Kalman filtering, scans the image row by row and computes the estimate at each pixel as a linear combination of the estimates at preceding, nearby pixels. The coefficients of the estimate that minimizes the expected squared error can be computed from the autocorrelation of the (ideal) image.

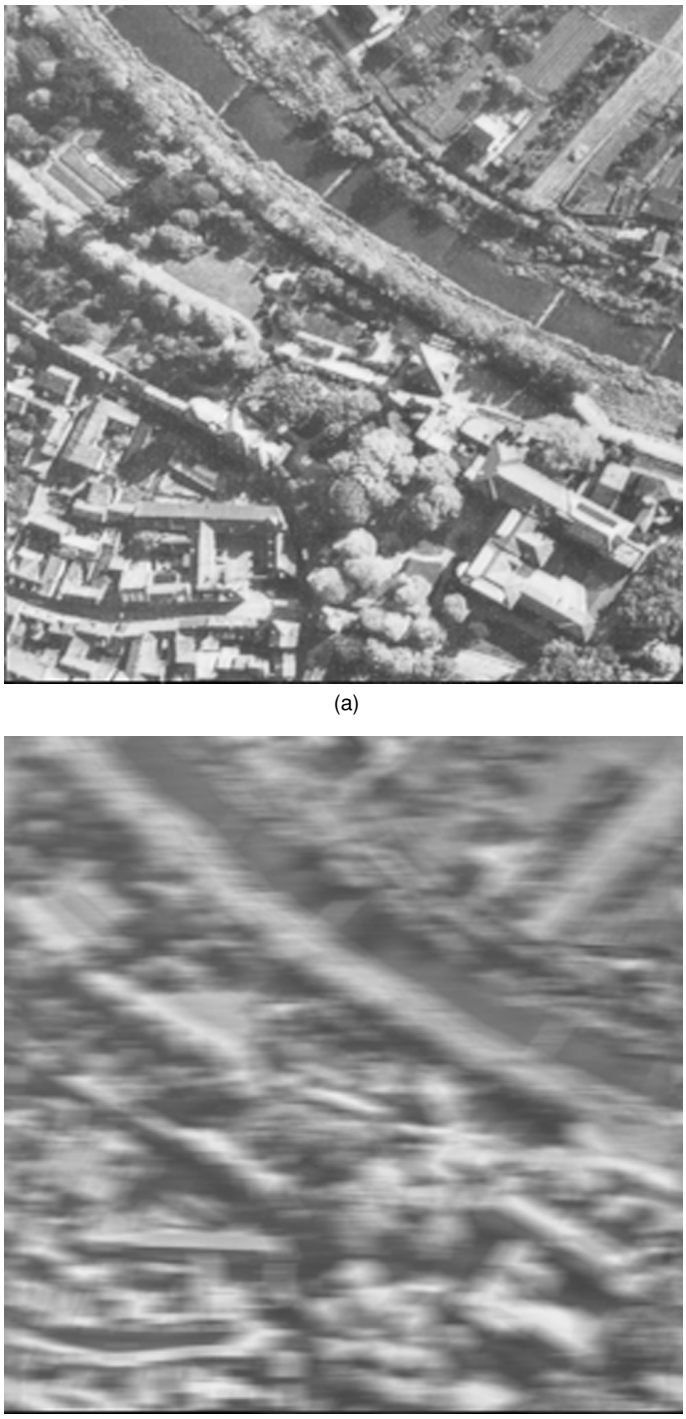
Recursive Kalman filters for image restoration are computationally intensive. Approximations such as reduced-update Kalman filters give very good results at a much lower computational complexity. The estimation-theoretic formulation of the image restoration problem using the Kalman filter has enabled investigation of more general cases, including blind restoration (where the unknown blur function is modeled and estimated along with the original image) and nonstationary image restoration (piecewise stationary regions are restored, with the filters appropriate for the regions being selected using a Markov chain).

In addition to recursive filters, other model-based estimation-theoretic approaches have been developed. For example, in the Wiener filter described above, one can use random field models (see Section III) to estimate the power spectra needed. Alternatively, one can use MRF models to characterize the degraded images and develop deterministic or stochastic estimation techniques that maximize the posterior probability density function.

A seminal approach that models the original image uisng a composite model, where stationary regions are represented using MRF models, and the discontinuities that separate the stationary regions are represented using "line processes," has yielded a new unified framework for handling a wide variety of problems in image estimation, restoration, surface reconstruction, and texture segmentation. The composite model, when used in conjunction with the MAP criterion, leads to nonconvex optimization problems. A class of stochastic search methods known as simulated annealing and its variants has enabled the solution of such optimization problems. Although these methods are computationally intensive, parallel hardware implementations of the annealing algorithms have taken the bite out of their computational complexity. An image restoration example is shown in Fig. 4.

VII. RECONSTRUCTION

A projection of an image is obtained by summing its gray levels along a family of parallel lines. In this section we show how an image can be (approximately) reconstructed from a sufficiently large set of its projections. This process of reconstruction from projections has important applications in reconstructing images of cross sections of a volume, given a set of X-ray images of the volume taken from different directions. In an X-ray image of an object, the ray striking the film at a given point has been attenuated by passing through the object; thus its strength is



(a)

(b)

FIGURE 4 Image restoration. (a) Original image; (b) blurred image; (c) restored image. [These images were provided by Prof. A. Katsaggelos of Northwestern University.]

proportional to the product of the attenuations produced by unit volumes of the object along its path. On a logarithmic scale, the product becomes a sum, so that an X-ray image (produced by a parallel beam of X rays) can be regarded as a projection of the object, and each row of the image can be regarded as a projection of a planar cross section of the object. Other methods of obtaining projections of a volume have been developed using radioactive tracers or ultrasound. An example of a cross section of the body reconstructed from a set of X rays is shown in Fig. 5.

(c)

FIGURE 4 *(continued)*

By the projection theorem for Fourier transforms, the 1-D transform of a projection of an image is equal to a cross section of the 2-D transform of the image. Specifically, let f_θ be the projection obtained by summing the gray levels of the image f along the family of lines in direction θ. Let F be the Fourier transform of f, and let $F^{\theta'}$ be the cross section of F along the line through the origin in direction $\theta' = \theta + \pi/2$. Then the Fourier transform of f_θ is just $F^{\theta'}$. This implies that if we have projections of f in many directions and we take their Fourier transforms, we obtain cross sections of the Fourier transform F along many lines through the origin. We can approximate F by interpolating from these cross sections and then reconstruct f by inverse Fourier transforming. Note that the

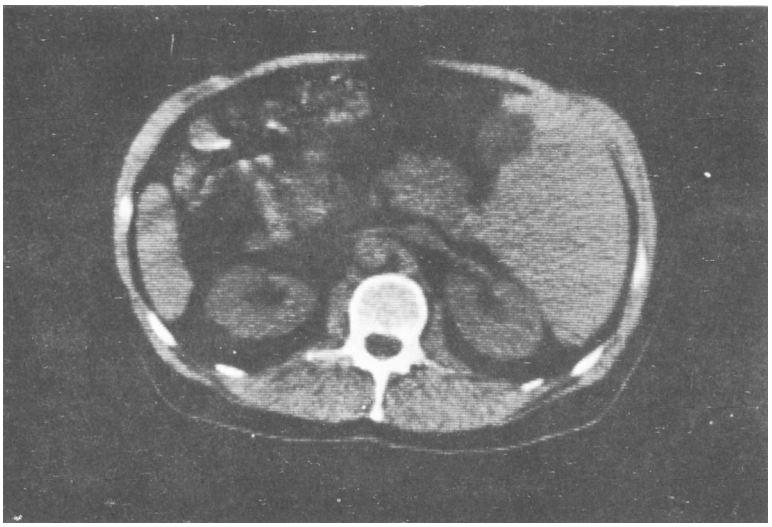


FIGURE 5 Reconstruction of a cross section of the human body from a set of X rays.

high-frequency components of f will not be reconstructed accurately using this method, since the cross-section data far from the origin are sparse.

Another approach to reconstruction from projections is based on "back-projection." The value v of a projection at a point is the sum of the values of the original image along a given line; to back-project, we divide v into many equal parts and distribute them along that line. If we do this for every projection, we obtain a highly blurred version of the original image. (To see this, suppose that the image contains a bright point P on a dark background; then P will give rise to a high value in each projection. When we back-project, these values will be spread along a set of lines that all intersect at P, giving rise to a high value at P and lower values along the lines radiating from it.) To obtain a more accurate reconstruction of the image, we combine back-projection with deblurring; we first filter the projections to precompensate for the blurring and then back-project them. This method of reconstruction is the one most commonly used in practice.

Each projection of an image f gives us a set of linear equations in the gray levels of f, since the value of a projection at a point is the sum of the gray levels along a line. Thus if we have a large enough set of projections of f, we can, in principle, solve a large set of linear equations to determine the original gray levels of f. Many variations of this algebraic approach to reconstruction have been formulated. (In principle, algebraic techniques can also be used in image deblurring; the gray level of a pixel in the blurred image is a weighted average of neighboring gray levels in the ideal image, so that the ideal gray levels can be found by solving a set of linear equations.)

Over the last 15 years, the emphasis in image reconstruction has been on introducing probabilistic or statistical principles in modeling noise and imaging mechanisms and deriving mathematically sound algorithms. As multimodal images (X ray, CT, MRI) are becoming increasingly available, registration of these images to each other and positioning of these images to an atlas have also become critical technologies. Visualization of reconstructed images and objects is also gaining attention.

VIII. MATCHING

There are many situations in which we want to "match" or compare two images with one another. The following are some common examples.

1. We can detect occurrences of a given pattern in an image by comparing the image with a "template" of the pattern. This concept has applications in pattern recognition, where we can use template matching to recognize known patterns, and also in navigation, where we can use landmark matching as an aid in determining the location from which an image was obtained.
2. Given two images of a scene taken from different positions, if we can identify points in the two images that correspond to a given scene point, we can determine the 3-D location of the scene point by triangulation. This process is known as stereomapping. The identification of corresponding points involves local matching of pieces of the two images.
3. Given two images of a scene taken (from the same position) at different times, we can determine the points at which they differ and analyze the changes that have taken place in the scene.

This section discusses template matching and how to measure the match or mismatch between two images, image registration, and stereomapping and range sensing. The analysis of time sequences of images is discussed in Section IX.

A. Template Matching

A standard measure of the "mismatch" or discrepancy between two images f and g is the sum of the absolute (or squared) differences between their gray levels at each point—for example, $\sum\sum(f-g)^2$. A standard measure of match is the correlation coefficient $\sum\sum fg/\sqrt{(\sum\sum f^2)(\sum\sum g^2)}$; by the Cauchy–Schwarz inequality, this is always ≤ 1 and is equal to 1 if f and g differ by a multiplicative constant. Thus to find places where the image f matches the template g, we can cross-correlate g with f (i.e., compute $\sum\sum fg$ for all shifts of g relative to f) and look for peaks in the correlation value. Note that by the convolution theorem for Fourier transforms, we can compute the cross-correlation by pointwise multiplying the Fourier transforms F and G^* (where the asterisk denotes the complex conjugate) and then inverse transforming.

The matched filter theorem states that if we want to detect matches between f and g by cross-correlating a filter h with f, and the criterion for detection is the ratio of signal power to expected noise power, then the best filter to use is the template g itself. Depending on the nature of f and the detection criterion, however, other filters may be better; for example, if f is relatively "smooth," better results are obtained by correlating the derivative of f with the derivative of g, or f with the second derivative of g.

Matching of derivatives generally yields sharper match peaks, but it is more sensitive to geometric distortion. To handle distortion, a good approach is first to match the

image with smaller (sub)templates, since these matches will be less sensitive to distortion, and then to look for combinations of these matches in approximately the correct relative positions.

Matching by cross-correlation is a computationally expensive process. We can reduce its cost by using inexpensive tests to determine positions in which a match is likely to be found, so that we need not test for a match in every possible position. One possibility is to match at a low resolution (i.e., at coarsely spaced points) and search for match peaks only in the vicinity of those points where the coarse match is good. An alternative is to match with a subtemplate and check the rest of the template only at those points where the subtemplate matches well. Note that if we use a very low resolution, or a very small subtemplate, there will be many false alarms, but if we use a relatively high resolution, or a relatively large subtemplate, the saving will not be significant.

B. Image Registration

Before we can match two or more images, they may have to be aligned to compensate for different acquisition viewpoints or times. For example, if we wish to combine information from CT, X-ray and MRI images, they have to be registered to a common coordinate frame before fusion or change detection can be attempted. Similar situations arise in image exploitation and automatic target recognition applications. There are also situations where instead of registering multiple images to each other, one needs to register multiple images, acquired at different times, to a 3-D model. These methods are referred to as model-supported positioning methods. Most traditional methods of image registration based on area correlation or feature matching can handle only minor geometric and photometric variations (typically, in images collected by the same sensor). In a multisensor context, however, the images to be registered may be of widely different types, obtained by disparate sensors with different resolutions, noise levels, and imaging geometries. The common or "mutual" information, which is the basis of automatic image registration, may manifest itself in a very different way in each image. This is because different sensors record different physical phenomena in the scene. For instance, an infrared sensor responds to the temperature distribution of the scene, whereas a radar responds to material properties such as dielectric constant, electrical conductivity, and surface roughness.

Since the underlying scene giving rise to the shared information is the same is all images of the scene, certain qualitative statements can be made about the manner in which information is preserved across multisensor data. Although the pixels corresponding to the same scene region may have different values depending on the sensor, pixel similarity and pixel dissimilarity are usually preserved. In other words, a region that appears homogeneous to one sensor is likely to appear homogeneous to another, local textural variations apart. Regions that can be clearly distinguished from one another in one image are likely to be distinguishable from one another in other images, irrespective of the sensor used. Although this is not true in all cases, it is generally valid for most types of sensors and scenes. Man-made objects such as buildings and roads in aerial imagery, and implants, prostheses, and metallic probes in medical imagery, also give rise to features that are likely to be preserved in multisensor images. Feature-based methods that exploit the information contained in region boundaries and in man-made structures are therefore useful for multisensor registration.

Feature-based methods traditionally rely on establishing feature correspondences between the two images. Such correspondence-based methods first employ feature matching techniques to determine corresponding feature pairs in the two images and then compute the geometric transformation relating them, typically using a least-squares approach. Their primary advantage is that the transformation parameters can be computed in a single step and are accurate if the feature matching is reliable. Their drawback is that they require feature matching, which is difficult to accomplish in a multisensor context and is computationally expensive, unless the two images are already approximately registered or the number of features is small.

Some correspondence-less registration methods based on moments of image features have been proposed, but these techniques, although mathematically elegant, work only if the two images contain exactly the same set of features. This requirement is rarely met in real images. Another proposed class of methods is based on the generalized Hough transform (GHT). These methods map the feature space into a parameter space, by allowing each feature pair to vote for a subspace of the parameter space. Clusters of votes in the parameter space are then used to estimate parameter values. These methods, although far more robust and practical than moment-based methods, have some limitations. Methods based on the GHT tend to produce large numbers of false positives. They also tend to be computationally expensive, since the dimensionality of the problem is equal to the number of transformation parameters.

Recently, methods similar in spirit to GHT-style methods, but employing a different search strategy to eliminate the problems associated with them, have been proposed. These methods first decompose the original transformation into a sequence of elementary stages. At each stage, the value of one transformation parameter

is estimated by a feature consensus mechanism in which each feature pair is allowed to estimate the value of the parameter that is consistent with it. The value of the parameter that is consistent with the most feature pairs is considered to be its best estimate. Using the concepts of parameter observability and parameter separability, it is possible in most cases to avoid the need for feature pairings; instead, aggregate properties of features determined from each image separately are used.

The global registration achieved by feature consensus should be sufficient for many applications such as those employing registration for determining a focus of attention. If more accurate global registration is needed, as in medical applications, the feature consensus result may be used as an initial condition for more elaborate schemes that use feature correspondence or multidimensional search, which require a good initial guess about the transformation parameters. Methods like deformable template matching can also be invoked for local refinement of the registration. An example of image registration is shown in Fig. 6.

C. Stereomapping and Range Sensing

Let P be a point in a scene and let P_1 and P_2 be the points corresponding to P in two images obtained from two known sensor positions. Let the "lens centers" in these two positions be L_1 and L_2, respectively. By the geometry of optical imaging, we know that P must lie on the line P_1L_1 (in space) and also on the line P_2L_2; thus its position in space is completely determined.

The difficulty in automating this process of stereomapping is that it is difficult to determine which pairs of image points correspond to the same scene point. (In fact, some scene points may be visible in one image but hidden in the other.) Given a point P_1 in one image, we can attempt to find the corresponding point P_2 in the other image by matching a neighborhood of P_1 with the second image. (We need not compare it with the entire image; since the camera displacement is known, P_2 must lie on a known line in the second image.) If the neighborhood used is too large, geometric distortion may make it impossible to find a good match; but if it is too small, there will be many false matches, and in a featureless region of the image, it will be impossible to find unambiguous (sharp) matches. Thus the matching approach will yield at best a sparse set of reasonably good, reasonably sharp matches. We can verify the consistency of these matches by checking that the resulting positions of the scene points in space lie on a smooth surface or a set of such surfaces. In particular, if the matches come from points that lie along an edge in an image, we can check that the resulting spatial positions lie on a smooth space curve.

(a)

FIGURE 6 Image registration. (a, b) The two images to be registered; (c) the registration result ("checkerboard" squares show alternating blocks of the two registered images.)

(b)

FIGURE 6 *(continued)*

(c)

FIGURE 6 *(continued)*

A strong theoretical basis for this approach evolved around the midseventies. Since then, advances have been made in interpolating the depth estimates obtained at positions of matched image features using surface interpolation techniques and hierarchical feature-based matching schemes, and dense estimates have been obtained using gray-level matching guided by simulated annealing. Although these approaches have contributed to a greater understanding of the problem of depth recovery using two cameras, much more tangible benefits have been reaped using a larger number of cameras. By arranging them in an array pattern, simple sum of squared difference-based schemes are able to produce dense depth estimates in real time. Using large numbers of cameras (in excess of 50), new applications in virtual reality, 3-D modeling, and computer-assisted surgery have become feasible.

Consistent with developments in multiscale analysis, stereo mapping has benefited from multiscale feature-based matching techniques. Also, simulated annealing and neural networks have been used for depth estimation using two or more images.

Another approach to determining the spatial positions of the points in a scene is to use patterned illumination. For example, suppose that we illuminate the scene with a plane of light Π, so that only those scene points that lie in Π are illuminated, and the rest are dark. In an image of the scene, any visible scene point P (giving rise to image point P_1) must lie on the line P_1L_1; since P must also lie in Π, it must be at the intersection of P_1L_1 and Π, so that its position in space is completely determined. We can obtain complete 3-D information about the scene by moving Π through a set of positions so as to illuminate every visible scene point, or we can use coded illumination in which the rays in each plane are distinctive (e.g., by their colors). A variety of "range sensing" techniques based on patterned illumination has been developed. Still another approach to range sensing is to illuminate the scene, one point at a time, with a pulse of light and to measure the time interval (e.g., the phase shift) between the transmitted and the reflected pulses, thus obtaining the range to that scene point directly, as in radar.

IX. IMAGE SEQUENCE ANALYSIS

Suppose that we are given a sequence of images (or frames) of the same scene taken at different times from the same position. By comparing the frames, we can detect changes that have taken place in the scene between one frame and the next. For example, if the frames are closely spaced in time and the changes are due to the motions of discrete objects in the scene, we can attempt to track the objects from frame to frame and so determine their motions. One way to do this is to match pieces of consecutive frames that contain images of a given object to estimate the displacement of that object. However, if the object is moving in three dimensions, the size and shape of its image may change from frame to frame, so that it may be difficult to find good matches.

During the early eighties, much effort was focused on estimating the motions and depths of features (points, lines, planar or quadric patches) using two or three frames. This problem was broken into two stages: establishing correspondences of features between frames and estimating motion and depth from the corresponding features using linear or nonlinear algorithms. However, the performance of these algorithms on real image sequences was not satisfactory.

To exploit the large numbers of frames in video image sequences, model-based recursive filtering approaches for estimating motion and depth were suggested in the mideighties. This approach led to applications of extended Kalman filters and their variants to problems in image sequence analysis. In general, model-based approaches enabled the use of multiple cameras, auxilary information such as inertial data, and statistical analysis. Successful applications of recursive filters to automobile navigation have been demonstrated. Figure 7 illustrates the use of recursive filters for feature tracking.

If the motion from frame to frame is not greater than the pixel spacing, then by comparing the space and time derivatives of the gray-level at a given point, in principle we can estimate the component of the image motion at that point in the direction of the gray-level gradient, but the component in the orthogonal (tangential) direction is ambiguous. Equivalently, when we look at a moving (straight) edge, we can tell how fast it is moving in the direction perpendicular to itself but not how fast it is sliding along itself. Unambiguous velocity estimates can be obtained at corners where two edges meet. Such estimates can be used as boundary conditions to find a smooth velocity field that agrees with the observed velocities at corners and whose

FIGURE 7 Feature tracking using recursive filtering.

components in the gradient direction agree with the observed components perpendicular to edges.

When a sequence of frames is taken by a moving sensor, there will be changes nearly everywhere, with the magnitude of the change at a given image point depending on the velocity of the sensor and on its distance from the corresponding scene point. The array of motion vectors at all points of the image is known as the optical flow field. An example of such a field is shown in Fig. 8. Different types of sensor motion (ignoring, for the moment, the motions of objects in the scene) give rise to different types of flow fields. Translational sensor motion perpendicular to the optical axis of the sensor simply causes each point of the image to shift, in the opposite direction, by an amount proportional to its distance from the sensor, so that the resulting image motion vectors are all parallel. Translation in other directions, on the other hand, causes the image to expand or shrink, depending on whether the sensor is approaching or receding from the scene; here the motion vectors all pass through a common point, the "focus of expansion" (or contraction). Thus if we know the translational sensor motion, we can compute the relative distance from the sensor to each point of the scene. (The distances are only relative because changes in absolute distance are indistinguishable from changes in the speed of the senser.) The effects of rotational sensor motion are more complex, but they can be treated independently of the translation effects.

Motion of a rigid body relative to the sensor gives rise to a flow field that depends on the motion and on the shape of the body. In principle, given enough (accurate) measurements of the flow in a neighborhood, we can determine both the local shape of the body and its motion.

Parallel to, and often independent of, the correspondence-based approach, optical flow-based structure and motion estimation algorithms have flourished for more than two decades. Although robust dense optical flow estimates are still elusive, the field has matured to the extent that systematic characterization and evaluation of optical flow estimates are now possible. Methods that use robust statistics, generalized motion models, and filters have all shown great promise. Significant work that uses directly observable flow ("normal flow") provides additional insight into the limitations of traditional approaches. An example of depth estimation using optical flow is shown in Fig. 8.

Segmentation of independently moving objects and dense scene structure estimation from computed flow have become mature research areas. New developments such as fast computation of depth from optical flow using fast Fourier transforms have opened up the possibility of real-time 3-D modeling. Another interesting accomplishment has been the development of algorithms

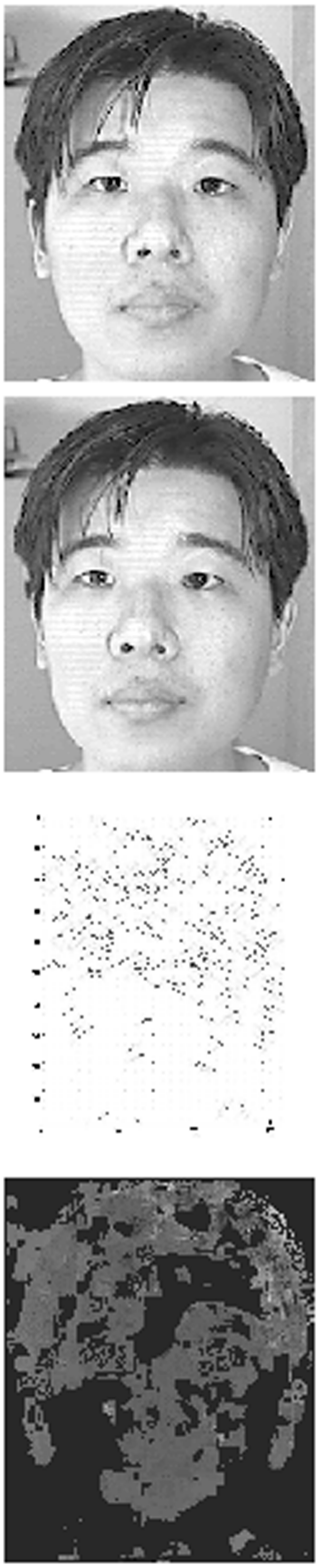


FIGURE 8 Depth estimation from optical flow. (a, b) Two frames of a video sequence; (c) computed flow field; (d) depth map; (e, f) views synthesized from the depth map.

for sensor motion stabilization and panoramic view generation.

Detection of moving objects, structure and motion estimation, and tracking of 2-D and 3-D object motion using contours and other features have important applications in surveillance, traffic monitoring, and automatic target recognition. Using methods ranging from recursive Kalman filters to the recently popular Monte Carlo Markov chain algorithms (known as CONDENSATION algorithms), extensive research has been done in this area. Earlier attempts were concerned only with generic tracking, but more recently, "attributed tracking," where one incorporates the color, identity, or shape of the object as part of the tracking algorithm, is gaining importance. Also, due to the impressive computing power that is now available, real-time tracking algorithms have been demonstrated.

X. RECOVERY

The gray level of an image at a given point P_1 is proportional to the brightness of the corresponding scene point P as seen by the sensor; P is the (usually unique) point on the surface of an object in the scene that lies along the line P_1L_1 (see Section VIII.C). The brightness of P depends in turn on several factors: the intensity of the illumination at P, the reflective properties of the surface S on which P lies, and the spatial orientation of S at P. Typically, if a light ray is incident on S at P from direction i, then the fraction r of the ray that emerges from S in a given direction e is a function of the angles θ_i and θ_e that i and e, respectively, make with the normal n to S at P. For example, in perfect specular reflection we have $r = 1$, if i and e are both coplanar with n and $\theta_i = \theta_e$, and $r = 0$ otherwise. In perfectly diffuse or Lambertian reflection, on the other hand, r depends only on θ, and not on θ_e; in fact, we have $r = p \cos\theta_i$, where p is a constant between 0 and 1.

If we could separate the effects of illumination, reflectivity, and surface orientation, we could derive 3-D information about the visible surfaces in the scene; in fact, the surface orientation tells us the rate at which the distance to the surface is changing, so that we can obtain distance information (up to a constant of integration) by integrating the orientation. The process of inferring scene illumination, reflectivity, and surface orientation from an image is called recovery (more fully, recovery of intrinsic scene characteristics from an image). Ideally, recovery gives us a set of digital image arrays in which the value of a pixel represents the value of one of these factors at the corresponding scene point; these arrays are sometimes called intrinsic images.

In this section we briefly describe several methods of inferring intrinsic scene characteristics from a single image. These methods are known as "shape from . . ." techniques, since they provide information about the 3-D shapes of the visible surfaces in the scene.

A. Shape from Shading

Suppose that a uniformly reflective surface is illuminated by a distant, small light source in a known position and is viewed by a distant sensor. Then the directions to the light source and to the sensor are essentially the same for two nearby surface points P and P'. Thus the change in surface brightness (and hence in image gray level) as we move from P to P' is due primarily to the change in direction of the surface normal. In other words, from the "shading" (i.e., the changes in gray level) in a region of the image, we can compute constraints on the changes in orientation of the corresponding surface in the scene. These constraints do not determine the orientation completely, but we can sometimes estimate the orientation of a smooth surface from shading information with the aid of boundary conditions (e.g., contours along which the surface orientation is known).

During the past two decades, significant strides have been made in recovering shape from shading. Using optimal control-theoretic principles, deeper understanding of the existence and uniqueness of solutions has become possible. Computational schemes using calculus of variations, multigrid methods, and linearization of the reflectance map have enabled practical implementations. Robust methods for simultaneously estimating the illuminant source direction and the surface shape have also been developed. Extensions to other imaging modalities, such as synthetic aperture radar and fusion of steropsis with shape from shading, have expanded the domain of applications.

If we can illuminate a scene successively from two (or more) directions, we can completely determine the surface orientation, for any surface that is visible in both images, by combining the constraints obtained from the shading in the two images. This technique is known as photometric stereo; an example is shown in Fig. 9.

An abrupt change in image gray level (i.e., an "edge" in an image) can result from several possible causes in the scene. It might be due to an abrupt change in illumination intensity; that is, it might be the edge of a shadow. It might also be due to an abrupt change in surface orientation (e.g., at an edge of a polyhedron), or it might be due to a change in reflectivity (e.g., an occluding edge where one surface partially hides another). Ideally, it should be possible to distinguish among these types of edges by careful analysis of the gray-level variations in their vicinities. For example, a shadow edge might not be very sharp, a convex orientation edge might have a highlight on it, and so on.

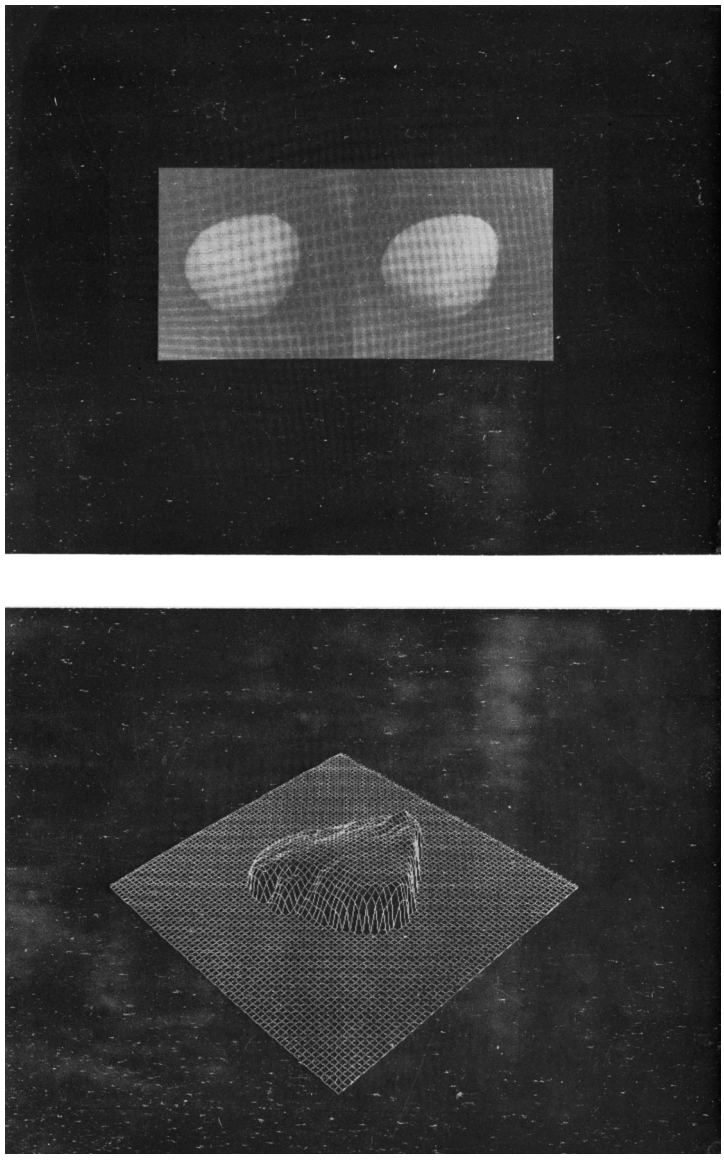


FIGURE 9 Photometric stereo.

B. Shape from Texture

Suppose that a surface is uniformly covered with an isotropic pattern or "texture." If the surface is flat and faces directly toward the sensor, its image will also be uniformly textured. On the other hand, if the surface is slanted, the patterns in the image will be foreshortened in the direction of the slant due to perspective distortion. Moreover, the scale of the image will change along the direction of the slant, since the distance from the sensor to the surface is changing; hence the sizes and spacings of the patterns will vary as we move across the image.

The slant of a uniformly textured surface can be inferred from anisotropies in the texture of the corresponding image region. For example, in an isotropic texture, the distribution of gray-level gradient directions should be uniform, but if the textured surface is slanted, the distribution of directions should have a peak in the direction of slant, due to the foreshortening effect, and the sharpness of the peak gives an indication of the steepness of the slant.

C. Shape from Shape

Various global assumptions can be used to infer surface orientation from the 2-D shapes of regions in an image. If the regions arise from objects of known shapes, it is straightforward to deduce the orientations of the objects from the shapes of their images, or their ranges from their image sizes. Even if the object shapes are unknown, certain inferences can be regarded as plausible. For example, if

an observed shape could be the projected image of a more symmetric or more compact slanted shape (e.g., an ellipse could be the projection of a slanted circle), one might assume that this is actually the case; in fact, human observers frequently make such assumptions. They also tend to assume that a continuous curve in an image arises from a continuous curve in space, parallel curves arise from parallel curves, straight lines arise from straight lines, and so on. Similarly, if two shapes in the image could arise from two congruent objects at different ranges or in different orientations, one tends to conclude that this is true.

A useful assumption about a curve in an image is that it arises from a space curve that is as planar as possible and as uniformly curved as possible. One might also assume that the surface bounded by this space curve has the least possible surface curvature (it is a "soap bubble" surface) or that the curve is a line of curvature of the surface. Families of curves in an image can be used to suggest the shape of a surface very compellingly; we take advantage of this when we plot perspective views of 3-D surfaces.

XI. SEGMENTATION

Images are usually described as being composed of parts (regions, objects, etc.) that have certain properties and that are related in certain ways. Thus an important step in the process of image description is segmentation, that is, the extraction of parts that are expected to be relevant to the desired description. This section reviews a variety of image segmentation techniques.

A. Pixel Classification

If a scene is composed of surfaces each of which has a constant orientation and uniform reflectivity, its image will be composed of regions that each have an approximately constant gray level. The histogram of such an image (see Section V.A) will have peaks at the gray levels of the regions, indicating that pixels having these levels occur frequently in the image, whereas other gray levels occur rarely. Thus the image can be segmented into regions by dividing the gray scale into intervals each containing a single peak. This method of segmentation is called thresholding.

Thresholding belongs to a general class of segmentation techniques in which pixels are characterized by a set of properties. For example, in a color image, a pixel can be characterized by its coordinates in color space—e.g., its red, green, and blue color components. If we plot each pixel as a point in color space, we obtain clusters of points corresponding to the colors of the surfaces. We can thus segment the image by partitioning the color space into regions each containing a single cluster. This general method of segmentation is called pixel classification.

Even in a black-and-white image, properties other than gray level can be used to classify pixels. To detect edges in an image (Section XI.B), we classify each pixel as to whether or not it lies on an edge by thresholding the rate of change of gray level in the neighborhood of that pixel; that is, we classify the pixel as edge or nonedge, depending on whether the rate of change is high or low. Similarly, we detect other features, such as lines or curves, by thresholding the degree to which the neighborhood of each pixel matches a given pattern or template.

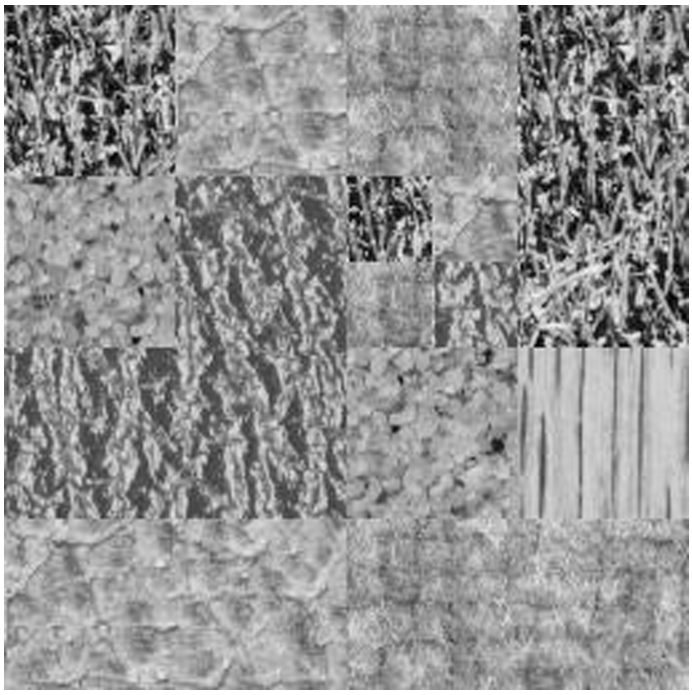
Properties derived from the neighborhoods of pixels—in brief, local properties—can also be used to segment an image into regions. For example, in a "busy" region the rate of change of gray level is often high, whereas in a "smooth" region it is always low. Suppose that we compute the rate of change at each pixel and then locally average it over a neighborhood of each pixel; then the average will be high for pixels in busy regions and low for pixels in smooth regions, so we can segment the image into such regions by thresholding the local average "busyness." We can use this method to segment an image into various types of differently "textured" regions by computing a set of local properties at each pixel and locally averaging their values; pixels belonging to a given type of textured region will give rise to a cluster in this "local property space."

Early pixel-based classification schemes made the assumption that adjacent pixels are uncorrelated. Since the mideighties, pixel classifications methods have tended to exploit the local correlation of pixels in single- or multiband images (often called "context" in the remote sensing literature). The use of MRFs and Gibbs distributions to characterize the local spatial/spectral correlation, combined with the use of estimation-theoretic concepts, has enabled the development of powerful pixel classification methods. An example of region segmentation is shown in Fig. 10.

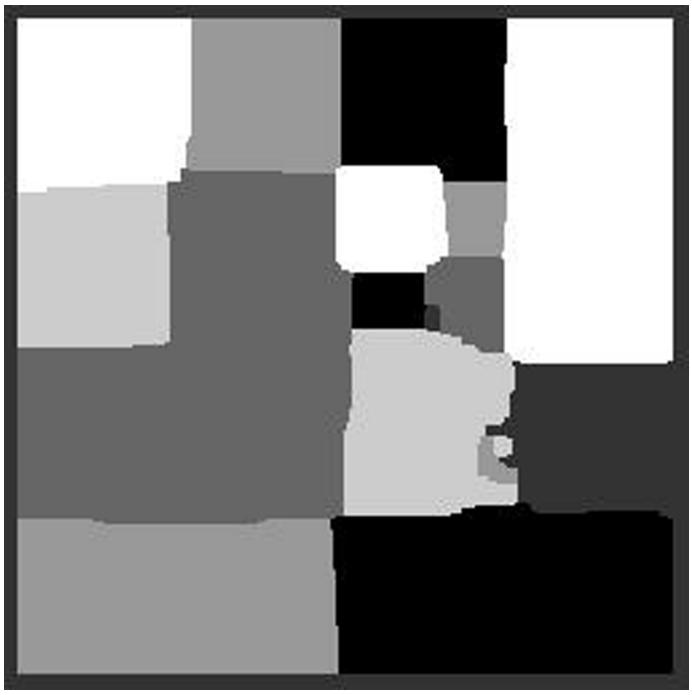
We have assumed up to now that the image consists of regions each of which has an approximately constant gray level or texture. If the scene contains curved surfaces, they will give rise to regions in the image across which the gray level varies; similarly, slanted textured surfaces in the scene will give rise to regions across which the texture varies. If we can estimate the orientations of the surfaces in the scene, as in Section X, we can correct for the effects of orientation.

B. Feature Detection

The borders of regions in an image are usually characterized by a high rate of change of gray level [or color or local property value(s)]. Thus we can detect region borders, or edges, in an image by measuring this rate of change (and thresholding it; see Section XI.A). For simplicity, we discuss only edges defined by gray-level changes, but similar



(a)



(b)

FIGURE 10 Texture segmentation. (a) Original texture mosaic; (b) segmented result.

remarks apply to other types of edges. The following are some standard methods of detecting edges.

1. Estimate the rate of change of gray level at each pixel P in two perpendicular directions using first-difference operators; let the estimates be f_1 and f_2. The gray-level gradient (i.e., greatest rate of change) at P is the vector whose magnitude is $\sqrt{f_1^2 + f_2^2}$ and whose direction is $\tan^{-1}(f_2/f_1)$.
2. Fit a polynomial surface to the gray levels in the neighborhood of P and use the gradient of the surface as an estimate of the gray level gradient.

3. Match a set of templates, representing the second derivatives of gray-level "steps" in various orientations, to the neighborhood of P (see Section VIII.A) and pick the orientation for which the match is best. (It turns out that convolving such a template with the image amounts to applying a first-difference operator at each pixel; thus this method amounts to computing first differences in many directions and picking the direction in which the difference is greatest.) Alternatively, fit a step function to the gray levels in the neighborhood of P and use the orientation and height of this step to define the orientation and contrast of the edge at P.
4. Estimate the Laplacian of the gray level at each pixel P. Since the Laplacian is a second-difference operator, it is positive on one side of an edge and negative on the other side; thus its zero-crossings define the locations of edges. (First differences should also be computed to estimate the steepnesses of these edges.)

Although the use of the Laplacian operator for edge detection has long been known, the idea of applying the Laplacian operator to a Gaussian smoothed image, using filters of varying sizes, stimulated theoretical developments in edge detection. In the mideighties, edge detectors that jointly optimize detection probability and localization accuracy were formulated and were approximated by directional derivatives of Gaussian-smoothed images. Fig. 11 shows an example of edge detection using this approach.

All these methods of edge detection respond to noise as well as to region borders. It may be necessary to smooth the image before attempting to detect edges; alternatively, one can use difference operators based on averages of blocks of gray levels rather than on single-pixel gray levels, so that the operator itself incorporates some smoothing. Various types of statistical tests can also be used to detect edges, based, for example, on whether the set of gray levels in two adjacent blocks comes from a single population or from two populations. Note that edges in an image can arise from many types of abrupt changes in the scene, including changes in illumination, reflectivity, or surface orientation; refinements of the methods described in this section would be needed to distinguish among these types.

To detect local features such as lines (or curves or corners) in an image, the standard approach is to match the image with a set of templates representing the second derivatives of lines (etc.) in various orientations. It turns out that convolving such a template with the image amounts to applying a second-difference operator at each pixel; thus this method responds not only to lines, but also to edges or to noise. Its response can be made more specific by applying the templates only when appropriate logical conditions are satisfied; for example, to detect a dark vertical line (on a light background) at the pixel P, we can require that P and its vertical neighbors each be darker than their horizontal neighbors. (Similar conditions can be incorporated when using template matching to detect edges.) To detect "coarse" features such as thick lines, similar templates and conditions can be used, but scaled up in size, that is, based on blocks of pixels rather than on single pixels.

It is usually impractical to use templates to detect global features such as long straight edges or lines, since too many templates would be needed to handle all possible positions and orientations. An alternative idea, called the Hough transform approach, is to map the image into a parameter space such that any feature of the given type gives rise to a peak in the space. For example, any straight line is characterized by its slope θ and its (shortest) distance ρ from the origin. To detect long straight lines in an image, we match it with local templates as described above; each time a match is found, we estimate its θ and ρ and plot them as a point in (θ, ρ) space. When all matches have been processed in this way, any long straight lines in the image should have given rise to clusters of points in (θ, ρ) space. Note that this technique detects sets of collinear line segments, whether or not they form a continuous line. A similar approach can be used to find edges or curves of any specific shape. Perceptual grouping of local features, using the Gestalt laws of grouping, provides an alternative approach when explicit shape information is not available.

C. Region Extraction

The segmentation techniques described in Sections XI.A and XI.B classify each pixel independently, based on its local properties; they are oblivious to whether the resulting set of pixels forms connected regions or continuous edges (etc.), and they therefore often give rise to noisy regions or broken edges. In this subsection we briefly discuss methods of segmentation that take continuity into account.

One approach is to require that the pixel classifications be locally consistent. If the classes represent region types and most of the neighbors of pixel P are of a given type, P should also be of that type. In edge (or curve) detection, if P lies on an edge in a given orientation, its neighbors in that direction should lie on edges in similar orientations. Once we have defined such consistency constraints, we can formulate the pixel classification task as one of optimization: Classify the pixels so as to maximize the similarity between each pixel and its class, while at the same time maximizing the consistencies between the classifications of neighboring pixels. Standard iterative methods can be

(a)

FIGURE 11 Edge detection. (a) Original image; (b) detected edges.

used to solve this optimization problem. If we are more concerned with consistency than with class similarity, we can simply reclassify pixels so as to make the results more consistent, thus reducing the "noise" in classified regions (or deleting isolated edge pixels), filling gaps in edges, and so on. The criteria of region consistency and edge consistency can, in principle, be combined to extract regions that are homogeneous and are surrounded by strong edges.

Another way of enforcing local continuity is to "grow" the regions by starting with (sets of) pixels whose classifications are clear-cut and extending them by repeatedly adding adjacent pixels that resemble those already classified. Similarly, to obtain continuous edges (or curves) we can start with strong-edge pixels and extend or "track" them in the appropriate directions. More generally, if we are given a set of region fragments or edge fragments, we can merge them into larger fragments based on similarity

(b)

FIGURE 11 *(continued)*

or consistency criteria. The results of these approaches will generally depend on the order in which the pixels or fragments are examined for possible incorporation or merging; it may be desirable to use search techniques (lookahead, backtracking) as an aid in making good choices.

We can also use global consistency criteria in segmenting an image; for example, we can require that the gray levels in a region be approximately constant (or, more generally, be a good fit to a planar or higher-order surface) or that a set of edge or line elements be a good fit to a straight line (or higher-order curve). To impose such a condition, we can start with the entire image, measure the fit, and split the image (e.g., into quadrants) if the fit is not good enough. The process can then be repeated for each quadrant until we reach a stage where no further splitting is needed; we have then partitioned the image into blocks on each of which the fit is good. Conversely, given such a partition, we may be able to merge some pairs of adjacent

parts and still have good fits to the merged parts. By using both splitting and merging, we can arrive at a partition such that the fit on each part is acceptable, but no two parts can be merged and still yield an acceptable fit. Note that by fitting surfaces to the gray levels we can, in principle, handle regions that arise from curved or shaded surfaces in the scene.

In many situations we have prior knowledge about the sizes and shapes of the regions that are expected to be present in the image, in addition to knowledge about their gray levels, colors, or textures. In segmentation by pixel classification we can make use of information about gray level (etc.) but not about shape. Using region-based methods of segmentation makes it easier to take such geometric knowledge into account; we can use region-growing, merging, or splitting criteria that are biased in favor of the desired geometries, or, more generally, we can include geometry-based cost factors in searching for an optimal partition. Cost criteria involving the spatial relations among regions of different types can also be used.

XII. GEOMETRY

In this section we discuss ways of measuring geometric properties of image subsets (regions or features) and of decomposing them into parts based on geometric criteria. We also discuss ways of representing image subsets exactly or approximately.

A. Geometric Properties

Segmentation techniques based on pixel classification can yield arbitrary sets of pixels as "segments." One often wants to consider the connected pieces of a subset individually, in order to count them, for example.

Let P and Q be pixels belonging to a given subset S. If there exists a sequence of pixels $P = P_0, P_1, \ldots, P_n = Q$, all belonging to S, such that, for each i, P_i is a neighbor of P_{i-1}, we say that P and Q are connected in S. The maximal connected subsets of S are called its (connected) components. We call S connected if it has only one component.

Let $\bar{S}$ be the complement of S. We assume that the image is surrounded by a border of pixels all belonging to $\bar{S}$. The component of $\bar{S}$ that contains this border is called the background of S; all other components of $\bar{S}$, if any, are called holes in S.

Two image subsets S and T are called adjacent if some pixel of S is a neighbor of some pixel of T. Let $S_1, \ldots, S_m$ be a partition of the image into subsets. We define the adjacency graph of the partition as the graph whose nodes are $S_1, \ldots, S_m$ and in which nodes S_i and S_j are joined by an arc if and only if S_i and S_j are adjacent. It can be shown that if we take $S_1, \ldots, S_m$ to be the components of S and of $\bar{S}$, where S is any subset of the image, then the adjacency graph of this partition is a tree.

The border of S consists of those pixels of S that are adjacent to (i.e., have neighbors in) $\bar{S}$. More precisely, if C is any component of S and D any component of $\bar{S}$, the D-border of C consists of those pixels of C (if any) that are adjacent to D. In general, if $S = C$ is connected, it has an outer border composed of pixels adjacent to the background, and it may also have hole borders composed of pixels adjacent to holes.

The area of S is the number of pixels in S. The perimeter of S is the total length of all its borders; it can be defined more precisely, for any given border, as the number of moves from pixel to pixel required to travel completely around the border and return to the starting point. The compactness of S is sometimes measured by a/p^2, where a is area and p is perimeter; this quantity is low when S has a jagged or elongated shape.

For any pixel of P of S, let $d(P)$ be the distance (in pixel units) from P to $\bar{S}$. The thickness of S is twice the maximum value of $d(P)$. If the area of S is large compared with its thickness, it is elongated.

S is called convex if for all pixels P, Q in S, the line segment $\overline{PQ}$ lies entirely within distance 1 of S. It is easy to show that, if S is convex, it must be connected and have no holes. The smallest convex set containing S is called the convex hull of S; we denote it $H(S)$. The difference set $H(S) - S$ is called the convex deficiency of S, and its components are called the concavities of S.

If surface orientation information is available, we can compute properties of the visible surfaces in the scene (not just of their projections in the image); for example, we can compute surface area rather than just region area. Many of the properties discussed above are very sensitive to spatial orientation and are of only limited value in describing scenes in which the orientations of the surfaces are not known.

B. Geometry-Based Decomposition

The concepts defined in Section XII.A provide us with many ways of defining new subsets of an image in terms of a given subset S. Examples of such subsets are the individual connected components of S, the holes in S, subsets obtained by "filling" the holes (e.g., taking the union of a component and its holes), the borders of S, the convex hull of S or its concavities, and so on. One can also "refine" a subset by discarding parts of it based on geometric criteria; for example, one can discard small

components or concavities (i.e., those whose areas are below some threshold).

An important class of geometric decomposition techniques is based on the concept of expanding and shrinking a set. [These operations are the same as min–max filtering (Section V.D) for the special case of a binary image.] Let $S^{(1)}$ be obtained from S by adding to it the border of $\bar{S}$ (i.e., adding all points of $\bar{S}$ that are adjacent to S), and let $S^{(2)}$, $S^{(3)}$, ... be defined by repeating this process. Let $S^{(-1)}$ be obtained from S by deleting its border (i.e., deleting all points of S that are adjacent to $\bar{S}$), and let $S^{(-2)}$, $S^{(-3)}$, ... be defined by repeating this process. If we expand S and then reshrink it by the same amount, that is, we construct $(S^{(k)})^{(-k)}$ for some k, it can be shown that the result always contains the original S; but it may contain other things as well. For example, if S is a cluster of dots that are less than $2k$ apart, expanding S will fuse the cluster into a solid mass, and reshrinking it will leave a smaller, but still solid mass that just contains the dots of the original cluster. Conversely, we can detect elongated parts of a set S by a process of shrinking and reexpanding. Specifically, we first construct $(S^{(-k)})^{(k)}$; it can be shown that this is always contained in the original S. Let S_k be the difference set $S - (S^{(-k)})^{(k)}$. Any component of S_k has a thickness of at most $2k$; thus if its area as large relative to k (e.g., $\geq 10k^2$), it must be elongated.

Another method of geometric decomposition makes use of shrinking operations that preserve the connectedness properties of S, by never deleting a pixel P if this would disconnect the remaining pixels of S in the neighborhood of P. Such operations can be used to shrink the components or holes of S down to single pixels or to shrink S down to a "skeleton" consisting of connected arcs and curves; the latter process is called thinning.

Still another approach to geometric decomposition involves detecting features of a set's border(s). As we move around a border, each step defines a local slope vector, and we can estimate the slope of the border by taking running averages of these vectors. By differentiating (i.e., differencing) the border slope, we can estimate the curvature of the border. This curvature will be positive on convex parts of the border and negative on concave parts (or vice versa); zero-crossings of the curvature correspond to points of inflection that separate the border into convex and concave parts. Similarly, sharp positive or negative maxima of the curvature correspond to sharp convex or concave "corners." It is often useful to decompose a border into arcs by "cutting" it at such corners or inflections. Similar remarks apply to the decomposition of curves. Many of the methods of image segmentation described in Section XI can be applied to border segmentation, with local slope vectors playing the role of pixel gray levels. Analogues of various image-processing techniques can also be applied to borders; for example, we can take the 1-D Fourier transform of a border (i.e., of a slope sequence) and use it to detect global periodicities in the border or filter it to smooth the border, or we can match a border with a template using correlation techniques.

C. Subset Representation

Any image subset S can be represented by a binary image in which a pixel has value 1 or 0 according to whether or not it belongs to S. This representation requires the same amount of storage space no matter how simple or complicated S is; for an $n \times n$ image, it always requires n^2 bits. In this section we describe several methods of representing image subsets that require less storage space if the sets are simple.

1. *Run length coding*. Each row of a binary image can be broken up into "runs" (maximal consecutive sequences) of 1's alternating with runs of 0's. The row is completely determined if we specify the value of the first run (1 or 0) and the lengths of all the runs, in the sequence. If the row has length n and there are only k runs, this run length code requires only $1 + k \log n$ bits, which can be much less than n if k is small and n is large.
2. *Maximal block coding*. The runs of 1's can be regarded as maximal rectangles of height 1 contained in the set S. We can obtain a more compact code if we allow rectangles of arbitrary height. Given a set of rectangles $R_1, \ldots, R_m$ whose union is S, we can determine S by specifying the positions and dimensions of these rectangles. As a specific example, for each pixel P in S, let S_P be the maximal upright square centered at P and contained in S. We can discard S_P if it is contained in S_Q for some $Q \neq P$. The union of the remaining S_P's is evidently S. Thus the set of centers and radii of these S_P's completely determines S; it is called the medial axis transformation of S, since the centers tend to lie on the "skeleton" of S.
3. *Quadtree coding*. Consider a binary image of size $2^n \times 2^n$. If it does not consist entirely of 1's or 0's, we divide it into quadrants; if any of these is not all 1's or all 0's, we divide it into subquadrants, and so on, until we obtain blocks that are all 1's or all 0's. The subdivision process can be represented by a tree of degree 4 (a quadtree); the root node represents the entire image, and each time we subdivide a block, we give its node four children representing its quadrants. If we specify the values (1 or 0) of the blocks represented by the leaves, the image is completely determined. The number of leaves will generally be

greater than the number of blocks in the medial axis transformation, but the quadtree has the advantage of concisely representing the spatial relations among the blocks by their positions in the tree, rather than simply storing them as an unsorted list.

4. *Contour coding*. Given any border of an image subset, we can define a border-following process that, starting from any pixel P of the border, successively visits all the pixels of that border, moving from neighbor to neighbor, and, finally, returns to P. The succession of moves made by this process, together with the coordinates of P, completely determines the border. (Each move can be defined by a code designating which neighbor to move to; the sequence of these codes is called the chain code of the border. Curves in an image can also be encoded in this way.) If all the borders of a set S are specified in this way, S itself is completely determined; that is, we can "draw" its borders and then "fill" its interior. This method of representing a set S is called contour coding. If S has only a few borders and their total length (i.e., the perimeter of S) is not too great, this representation can be more compact than the representation of S by a binary image.

For all these representations, efficient algorithms have been developed for computing geometric properties directly from the representation, for computing the representations of derived subsets (unions, intersections, etc.) directly from the representations of the original subsets, and for converting one representation to another.

The representations defined above are all exact; they completely determine the given subset. We can also define approximations to an image, or to an image subset, using similar methods. For example, we can use maximal blocks whose values are only approximately constant; compare the method of segmentation by splitting described in Section XI.C. Similarly, we can approximate a border or curve—for example, by a sequence of straight-line segments. We can approximate a medial axis by a set of arcs and a radius function defined along each arc; a shape defined by a simple arc and its associated function is called a generalized ribbon. It is often useful to compute a set of approximations of different degrees of accuracy; if these approximations are refinements of one another, they can be stored in a tree structure.

Representations analogous to those described here can be defined for 3-D objects. Such 3-D representations are needed in processing information about surfaces (e.g., obtained from stereo, range sensing, or recovery techniques) or in defining the objects in a scene that we wish to recognize in an image.

XIII. DESCRIPTION

The goal of image analysis is usually to derive a description of the scene that gave rise to the image, in particular, to recognize objects that are present in the scene. The description typically refers to properties of and relationships among objects, surfaces, or features that are present in the scene, and recognition generally involves comparing this descriptive information with stored "models" for known classes of objects or scenes. This section discusses properties and relations, their representation, and how they are used in recognition.

A. Properties and Relations

In Section XII.A we defined various geometric properties of and relationships among image parts. In this section we discuss some image properties that depend on gray level (or color), and we also discuss the concept of invariant properties.

The moments of an image provide information about the spatial arrangement of the image's gray levels. If $f(x, y)$ is the gray level at (x, y), the (i, j) moment m_{ij} is defined as $\sum\sum x^i y^i f(x, y)$ summed over the image. Thus m_{00} is simply the sum of all the gray levels. If we think of gray level as mass, $(m_{10}/m_{00}, m_{01}/m_{00})$ are the coordinates of the centroid of the image. If we choose the origin at the centroid and let $\bar{m}_{ij}$ be the (i, j) central moment relative to this origin, then $\bar{m}_{10} = \bar{m}_{01} = 0$, and the central moments of higher order provide information about how the gray levels are distributed around the centroid. For example, $\bar{m}_{20}$ and $\bar{m}_{02}$ are sensitive to how widely the high gray levels are spread along the x and y axes, whereas $\bar{m}_{30}$ and $\bar{m}_{03}$ are sensitive to the asymmetry of these spreads. The principal axis of the image is the line that gives the best fit to the image in the least-squares sense; it is the line through the centroid whose slope $\tan\theta$ satisfies

$$\tan^2\theta + \frac{\bar{m}_{20} - \bar{m}_{02}}{\bar{m}_{11}} \tan\theta - 1 = 0.$$

Moments provide useful information about the layout or shape of an image subset that contrasts with its complement; they can be computed without the need to segment the subset explicitly from the rest of the image. (Many of the geometric properties defined in Section XII can also be defined for "fuzzy" image subsets that have not been explicitly segmented.)

If the gray levels in an image (or region) are spatially stationary (intuitively, the local pattern of gray levels is essentially the same in all parts of the region), various statistical measures can be used to provide information about the texture of the region. The "coarseness" of the texture can can be measured by how slowly the image's

autocorrelation drops off from its peak at zero displacement (or, equivalently, by how rapidly the image's Fourier power spectrum drops off from its peak at zero frequency), and the "directionality" of the texture can be detected by variation in the rate of dropoff with direction. Strong periodicities in the texture give rise to peaks in the Fourier power spectrum. More information is provided by the second-order probability density of gray levels, which tells us how often each pair of gray levels occurs at each possible relative displacement. (The first-order gray level probability density tells us about the population of gray levels but not about their spatial arrangement.) Some other possible texture descriptors are statistics of gray-level run lengths, gray-level statistics after various amounts and types of filtering, or the coefficients in a least-squares prediction of the gray level of a pixel from those of its neighbors. Information about the occurrence of local patterns in a texture is provided by first-order probability densities of various local property values (e.g., degrees of match to templates of various types). Alternatively, one can detect features such as edges in a texture, or segment the texture into microregions, and compute statistics of properties of these features or microregions (e.g., length, curvature, area, elongatedness, average gray level) and of their spatial arrangement. Overall descriptions of region texture are useful primarily when the regions are the images of flat surfaces oriented perpendicularly to the line of sight; in images of curved or slanted surfaces, the texture will not usually be spatially stationary (see Section X.B).

The desired description of an image is often insensitive to certain global transformations of the image; for example, the description may remain the same under stretching or shrinking of the gray scale (over a wide range) or under rotation in the image plane. This makes it desirable to describe the image in terms of properties that are invariant under these transformations. Many of the geometric properties discussed in Section XII are invariant under various geometric transformations. For example, connectivity properties are invariant under arbitrary "rubber-sheet" distortion; convexity, elongatedness, and compactness are invariant under translation, rotation, and magnification; and area, perimeter, thickness, and curvature are invariant under translation and rotation. (This invariance is only approximate, because the image must be redigitized after the transformation.) The autocorrelation and Fourier power spectrum of an image are invariant under (cyclic) translation, and similar transforms can be defined that are invariant under rotation or magnification. The central moments of an image are invariant under translation, and various combinations of moments can be defined that are invariant under rotation or magnification. It is often possible to normalize an image, that is, to transform it into a "standard form," such that all images differing by a given type of transformation have the same standard form; properties measured on the normalized image are thus invariant to transformations of that type. For example, if we translate an image so its centroid is at the origin and rotate it so its principal axis is horizontal, its moments become invariant under translation and rotation. If we flatten an image's histogram, its gray-level statistics become independent of monotonic transformations of the grayscale; this is often done in texture analysis. It is much more difficult to define properties that are invariant under 3-D rotation, since as an object rotates in space, the shape of its image can change radically, and the shading of its image can change nonmonotonically.

In an image of a 2-D scene, many types of relationships among image parts can provide useful descriptive information. These include relationships defined by the relative values of properties ("larger than," "darker than," etc.) as well as various types of spatial relations ("adjacent to," "surrounded by," "between," "near," "above," etc.). It is much more difficult to infer relationships among 3-D objects or surfaces from an image, but plausible inferences can sometimes be made. For example, if in the region on one side of an edge there are many edges or curves that abruptly terminate where they meet the edge, it is reasonable to infer that the surface on that side lies behind the surface on the other side and that the terminations are due to occlusion. In any case, properties of and relationships among image parts imply constraints on the corresponding object parts and how they could be related and, thus, provide evidence about which objects could be present in the scene, as discussed in the next subsection.

B. Relational Structures and Recognition

The result of the processes of feature extraction, segmentation, and property measurement is a collection of image parts, values of properties associated with each part, and values of relationships among the parts. Ideally, the parts should correspond to surfaces in the scene and the values should provide information about the properties of and spatial relationships among these surfaces. This information can be stored in the form of a data structure in which, for example, nodes might represent parts; there might be pointers from each node to a list of the properties of that part (and, if desired, to a data structure such as a chain code or quadtree that specifies the part as an image subset) and pointers linking pairs of nodes to relationship values.

To recognize an object, we must verify that, if it were present in the scene, it could give rise to an image having the observed description. In other words, on the basis of our knowledge about the object (shape, surface properties, etc.) and about the imaging process (viewpoint, resolution,

etc.), we can predict what parts, with what properties and in what relationships, should be present in the image as a result of the presence of the object in the scene. We can then verify that the predicted description agrees with (part of) the observed description of the image. Note that if the object is partially hidden, only parts of the predicted description will be verifiable. The verification process can be thought of as being based on finding a partial match between the observed and predicted data structures. Note, however, that this is not just a simple process of (say) labeled graph matching; for example, the predicted image parts may not be the same as the observed parts, due to errors in segmentation.

If the viewpoint is not known, recognition becomes much more difficult, because the appearance of an object (shape, shading, etc.) can vary greatly with viewpoint. A brute-force approach is to predict the appearance of the object from many different viewpoints and match the observed description with all of these predictions. Alternatively, we can use a constraint analysis approach such as the following. For any given feature or region in the image, not every feature or surface of the object could give rise to it (e.g., a compact object can never give rise to an elongated region in the image), and those that can give rise to it can do so only from a limited set of viewpoints. If a set of image parts could all have arisen from parts of the same object seen from the same viewpoint, we have strong evidence that the object is in fact present.

C. Models

In many situations we need to recognize objects that belong to a given class, rather than specific objects. In principle, we can do this if we have a "model" for the class, that is, a generic description that is satisfied by an object if and only if it belongs to the class. For example, such a model might characterize the objects as consisting of sets of surfaces or features satisfying certain constraints on their property and relationship values. To recognize an object as belonging to that class, we must verify that the observed configuration of image parts could have arisen from an object satisfying the given constraints.

Unfortunately, many classes of objects that humans can readily recognize are very difficult to characterize in this way. Object classes such as trees, chairs, or even handprinted characters do not have simple generic descriptions. One can "characterize" such classes by simplified, partial descriptions, but since these descriptions are usually incomplete, using them for object recognition will result in many errors. Even the individual parts of objects are often difficult to model; many natural classes of shapes (e.g., clouds) or of surface textures (e.g., tree bark) are themselves difficult to characterize.

We can define relatively complex models hierarchically by characterizing an object as composed of parts that satisfy given constraints, where the parts are in turn composed of subparts that satisfy given constraints, and so on. This approach is used in syntactic pattern recognition, where patterns are recognized by the detection of configurations of "primitive" parts satisfying given constraints, then configurations of such configurations, and so on. Unfortunately, even though the models defined in this way can be quite complex, this does not guarantee that they correctly characterize the desired classes of objects.

Models are sometimes defined probabilistically, by the specification of probability densities on the possible sets of parts, their property values, and so on. This allows recognition to be probabilistic, in the sense that we can use Bayesian methods to estimate the probability that a given observed image configuration arose from an object belonging to a given class. This approach is used in statistical pattern recognition. Unfortunately, making a model probabilistic does not guarantee its correctness, and in any case, the required probability densities are usually very difficult to estimate.

Real-world scenes can contain many types of objects. A system capable of describing such scenes must have a large set of models available to it. Checking these models one by one against the image data would be very time-consuming. The models should be indexed so the appropriate ones can be rapidly retrieved, but not much is known about how to do this.

D. Knowledge-Based Recognition Systems

Nearly all existing image analysis systems are designed to carry out fixed sequences of operations (segmentation, property measurement, model matching, etc.) on their input images. The operations are chosen by the system designer on the basis of prior knowledge about the given class of scenes. For example, we choose segmentation operations that are likely to extract image parts corresponding to surfaces in the scene, we measure properties that are relevant to the desired description of the scene, and so on.

A more flexible type of system would have the capacity to choose its own operations on the basis of knowledge about the class of scenes (and the imaging process) and about how the operations are expected to perform on various types of input data. At each stage and in each part of the image, the system would estimate the expected results of various possible actions, based on its current state of knowledge, and choose its next action to have maximum utility, where utility is a function of the cost of the action and the informativeness of the expected results.

Many examples of knowledge-based reocgnition systems have been demonstrated in the image understanding

and computer vision literature. A more recent idea is to design a supervisor for the knowledge-based system, so that by evaluating the system's current state (quality of image, required speed, etc.), the best possible algorithm, with optimal choice of parameters, can be chosen from among the available options. Designing such a system, however, requires active participation by the user.

XIV. ARCHITECTURES

Image processing and analysis are computationally costly because they usually involve large amounts of data and complex sets of operations. Typical digital images consist of hundreds of thousands of pixels, and typical processing requirements may involve hundreds or even thousands of computer operations per pixel. If the processing must be performed rapidly, conventional computers may not be fast enough. For this reason, many special-purpose computer architectures have been proposed or built for processing or analyzing images. These achieve higher processing speeds by using multiple processors that operate on the data in parallel.

Parallelism could be used to speed up processing at various stages; for example, in image analysis, one could process geometric representations in parallel or match relational structures in parallel. However, most of the proposed approaches have been concerned only with parallel processing of the images themselves, so we consider here only operations performed directly on digital images.

The most common class of image operations comprises local operations, in which the output value at a pixel depends only on the input values of the pixel and a set of its neighbors. These include the subclass of point operations, in which the output value at a pixel depends only on the input value of that pixel itself. Other important classes of operations are transforms, such as the discrete Fourier transform, and statistical computations, such as histogramming or computing moments; in these cases each output value depends on the entire input image. Still another important class consists of geometric operations, in which the output value of a pixel depends on the input values of some other pixel and its neighbors. We consider in this section primarily local operations.

A. Pipelines

Image processing and analysis often require fixed sequences of local operations to be performed at each pixel of an image. Such sequences of operations can be performed in parallel using a pipeline of processors, each operating on the output of the preceding one. The first processor performs the first operation on the image, pixel by pixel. As soon as the first pixel and its neighbors have been processed, the second processor begins to perform the second operation, and so on. Since each processor has available to it the output value of its operation at every pixel, it can also compute statistics of these values, if desired.

Let t be the longest time required to perform an operation at one pixel, and let kt be the average delay required, after an operation begins, before the next operation can start. If there are m operations and the image size is $n \times n$, the total processing time required is then $n^2t + (m-1)kt$. Ordinarily n is much greater than m or k, so that the total processing time is not much greater than that needed to do a single operation (the slowest one).

Pipelines can be structured in various ways to take advantage of different methods of breaking down operations into suboperations. For example, a local operation can sometimes be broken into stages, each involving a different neighbor, or can be broken up into individual arithmetical or logical operations. Many pipeline image-processing systems have been designed and built.

B. Meshes

Another way to use parallel processing to speed up image operations is to divide the image into equal-sized blocks and let each processor operate on a different block. Usually the processors will also need some information from adjacent blocks; for example, to apply a local operation to the pixels on the border of a block, information about the pixels on the borders of the adjacent blocks is needed. Thus processors handling adjacent blocks must communicate. To minimize the amount of communication needed, the blocks should be square, since a square block has the least amount of border for a given area. If the image is $n \times n$, where $n = rs$, we can divide it into an $r \times r$ array of square blocks, each containing $s \times s$ pixels. The processing is then done by an $r \times r$ array of processors, each able to communicate with its neighbors. Such an array of processors is called a mesh-connected computer (or mesh, for short) or, sometimes, a cellular array. Meshes are very efficient at performing local operations on images. Let t be the time required to perform the operation at one pixel, and let c be the communication time required to pass a pixel value from one processor to another. Then the total processing time required is $4cs + ts^2$. Thus by using r^2 processors we have speeded up the time required for the operation from tn^2 to about ts^2, which is a speedup by nearly a factor of r^2. As r approaches n (one processor per pixel), the processing time approaches t and no longer depends on n at all.

For other types of operations, meshes are not quite so advantageous. To compute a histogram, for example, each processor requires time on the order of s^2 to count the

values in its block of the image, but the resulting counts must still be transmitted to one location (e.g., to the processor in the upper left corner) so they can be added together. The transmission requires on the order of r relay stages, since the data must be communicated from one processor to another. Thus as r approaches n, the time required to compute a histogram becomes on the order of n. This is much faster than the single-processor time, which is on the order of n^2, but it is still not independent of n. It can be shown that the time required on a mesh to compute a transform or to perform a geometric operation is also on the order of n as r approaches n.

The speed of a mesh is limited in performing nonlocal operations, because information must be transmitted over long distances, and the processors in the mesh are connected only to their neighbors. Much faster processing times, on the order of $\log n$, can be achieved by the use of a richer interprocessor connection structure, based, for example, on a tree or a hypercube. Meshes augmented in this way allow a wide variety of operations to be performed on images at very high speeds.

C. Recent Trends

Given the impressive advances in the computing power of personal computers and other workstations, the role of computer architectures in image processing and analysis has changed. One can now cluster a set of PCs to realize the computational power of a supercomputer at a much lower cost. This, together with the ability to connect cameras and drivers to a PC, has made image processing feasible at every desk. Important advances are also being made in implementing image processing and understanding algorithms on embedded processors, field-programmable gate arrays, etc. Also, more emphasis is being given to low-power implementations so that battery life can be extended.

XV. SUMMARY

Given the advances that have been made over the last 15 years in image processing, analysis, and understanding, this article has been very difficult for us to revise. After completing the revision, we paused to ponder what has been achieved in these fields over the last 50 years. Based on our combined perspectives, we have formulated the following observations that encapsulate the excitement that we still feel about the subject.

1. Sensors—Instead of the traditional single camera in the visible spectrum, we now have sensors that can cover a much wider spectrum and can perform simple operations on the data.
2. Algorithms—Starting from heuristic, ad hoc concepts, we have reached a stage where statistics, mathematics, and physics play key roles in algorithm development. Wherever possible, existence and uniqueness issues are being addressed. Instead of the traditional passive mode in which algorithms simply process the data, we can now have the algorithms control the sensors so that appropriate data can be collected, leading to an active vision paradigm.
3. Architectures—In the early seventies it took considerable effort and money to set up an image processing/computer vision laboratory. Due to the decreasing costs of computers and memory and their increasing power and capacity, we are now not far away from the day when palm processing of images will be possible.
4. Standards—Advances in JPEG, JPEG-2000, and various MPEG standards have resulted in at least some image processing topics (image compression, video compression) becoming household concepts.
5. Applications—In addition to traditional military applications, we are witnessing enormous growth in applications in medical imaging, remote sensing, document processing, internet imaging, surveillance, and virtual reality. The depth and breadth of the field keep growing!

SEE ALSO THE FOLLOWING ARTICLES

COLOR SCIENCE • COMPUTER ALGORITHMS • COMPUTER ARCHITECTURE • IMAGE-GUIDED SURGERY • IMAGE RESTORATION, MAXIMUM ENTROPY METHODS

BIBLIOGRAPHY

Aloimonos, Y. (ed.) (1993). "Active Perception," Lawrence Erlbaum, Hillsdale, NJ.

Ballard, D. H., and Brown, C. M. (1982). "Computer Vision," Prentice–Hall, Englewood Cliffs, NJ.

Blake, A., and Zisserman, A. (1987). "Visual Reconstruction," MIT Press, Cambridge, MA.

Brady, J. M. (ed.) (1981). "Computer Vision," North-Holland, Amsterdam.

Chellappa, R., and Jain, A. K. (1993). "Markov Random Fields: Theory and Application," Academic Press, San Diego, CA.

Chellappa, R., Girod, B., Munson, D., Jr., Tekalp, M., and Vetterli, M. (1998). "The Past, Present and Future of Image and Multidimensional Signal Processing." *IEEE Signal Process. Mag.* **15**(2), 21–58.

Cohen, P. R., and Feigenbaum, E. A. (eds.) (1982). "Handbook of Artificial Intelligence," Vol. III, Morgan Kaufmann, Los Altos, CA.

Ekstrom, M. P. (ed.) (1984). "Digital Image Processing Techniques," Academic Press, Orlando, FL.

Faugeras, O. D. (ed.) (1983). "Fundamentals in Computer Vision—An Advanced Course," Cambridge University Press, New York.

Faugeras, O. D. (1996). "Three-Dimensional Computer Vision—A Geometric Viewpoint," MIT Press, Cambridge, MA.
Gersho, A., and Gray, R. M. (1992). "Vector Quantization and Signal Compression," Kluwer, Boston.
Grimson, W. E. L. (1982). "From Images to Surfaces," MIT Press, Cambridge, MA.
Grimson, W. E. L. (1990). "Object Recognition by Computer—The Role of Geometric Constraints," MIT Press, Cambridge, MA.
Hanson, A. R., and Riseman, E. M. (eds.) (1978). "Computer Vision Systems," Academic Press, New York.
Horn, B. K. P. (1986). "Robot Vision," MIT Press, Cambridge, MA, and McGraw–Hill, New York.
Horn, B. K. P., and Brooks, M. (eds.) (1989). "Shape from Shading," MIT Press, Cambridge, MA.
Huang, T. S. (ed.) (1981a). "Image Sequence Analysis," Springer, Berlin.
Huang, T. S. (ed.) (1981b). "Two-Dimensional Digital Signal Processing," Springer, Berlin.
Jain, A. K. (1989). "Fundamentals of Digital Image Processing," Prentice–Hall, Englewood Cliffs, NJ.
Kanatani, K. (1990). "Group-Theoretic Methods in Image Understanding," Springer, Berlin.
Koenderink, J. J. (1990). "Solid Shape," MIT Press, Cambridge, MA.
Marr, D. (1982). "Vision—A Computational Investigation into the Human Representation and Processing of Visual Information," Freeman, San Francisco.
Mitchell, J. L., Pennebaker, W. B., Fogg., C. E., and LeGall, D. J. (1996). "MPEG Video Compression Standard," Chapman and Hall, New York.
Pavlidis, T. (1982). "Algorithms for Graphics and Image Processing," Computer Science Press, Rockville, MD.
Pennebaker, W. B., and Mitchell, J. L. (1993). "JPEG Still Image Compression Standard," Van Nostrand Reinhold, New York.
Pratt, W. K. (1991). "Digital Image Processing," Wiley, New York.
Rosenfeld, A., and Kak, A. C. (1982). "Digital Picture Processing," 2nd ed., Academic Press, Orlando, FL.
Ullman, S. (1979). "The Interpretation of Visual Motion," MIT Press, Cambridge, MA.
Weng, J., Huang, T. S., and Ahuja, N. (1992). "Motion and Structure from Image Sequences," Springer, Berlin.
Winston, P. H. (ed.) (1975). "The Psychology of Computer Vision," McGraw–Hill, New York.
Young, T. Y., and Fu, K. S. (eds.) (1986). "Handbook of Pattern Recognition and Image Processing," Academic Press, Orlando, FL.
Zhang, Z., and Faugeras, O. D. (1992). "3D Dynamic Scene Analysis—A Stereo-Based Approach," Springer, Berlin.