MULTISCALE IMAGE UNDERSTANDING

by

Charles R. Dyer

Computer Sciences Technical Report #679

December 1986

Multiscale Image Understanding

Charles R. Dyer

Computer Sciences Department
University of Wisconsin
Madison, WI 53706

Abstract

This paper reviews methods for understanding multiscale (also called multiresolution) image descriptions. We include work related to the construction and analysis of image representations which make explicit properties of edges, shape and texture at multiple scales. In addition we present two applications of multiscale techniques for model-based object recognition and texture segmentation.

This work was supported in part by the National Science Foundation under Grant DCR-8520870.

Table of Contents

1. Introduction

Scenes of the world contain objects of many sizes containing features of many sizes. Moreover, objects can be viewed at various distances. Consequently, the images we see contain features at many different scales. For example, the coat of a zebra can be described at several different, almost independent, scales — at the finest level the individual hairs are visible, at the next level collections of hairs form surface patterns, and at a still coarser level the stripes are seen. Choosing the correct scale for analyzing image features is therefore critical for recovering a complete physical interpretation of the objects in a scene.

Three physical properties which are affected by scale are edges, shape and texture. In each case the measurement and description of the property is a function of the image in a neighborhood or window centered on a point of interest: The spatial locality of brightness changes in an image determines the scale of the edges that we perceive. Similarly, the perception of shape is influenced by the scale at which we describe local curvature. Texture is also a scale-dependent property since it depends on brightness variability as a function of area.

For example, the edges in a scene are detected by locating abrupt changes in brightness in an image. But there are many different types of edges which can physically occur in a scene. For example, edges occur when there is a change in incident illumination (i.e. the boundary of a shadow cast on a smooth surface), when the surface orientation is normal to the line of sight (i.e. where the surface turns away from the viewer), or when there is a change in the albedo of the surface (i.e. the reflectance properties of the surface change). Each edge type produces its own brightness profile in terms of resolution (width), contrast (height) and acutance (shape). In particular, the resolution of the edge determines the locality of the brightness change. In finely textured surfaces, the detection of "microedges" is important in order to distinguish the texture primitives. (For example, the individual hairs on the zebra's coat.) Consequently, abrupt brightness changes between pairs of adjacent pixels are significant in such areas of the image. To distinguish the boundary between two coarsely textured regions, a "macroedge" could be defined between a pair of adjacent pixels if the two points are at the borders of two large regions containing different average gray levels (even if the two pixels have the same gray level). (For example, at the boundaries between stripes on the zebra.) At "shadow edges" the change in brightness occurs very gradually over a wide spatial area. Thus these three types of edges are visible at three distinct scales or resolutions in an image — the microedges are fine resolution features, the macroedges are coarse resolution and the shadow edges are intermediate resolution. Furthermore, other edges can only be detected using a property other than brightness; for example, the coarse bands visible in a herringbone pattern are demarcated where lines change direction.

The difficulty of detecting significant features at the appropriate scale is compounded by the fact that features at one scale can make the detection of nearby features at a different scale more difficult. In addition, there is no basis for choosing *a priori* what the best scale of measurement is either for an entire image or for specific areas within an image. Finally, the significance of a feature of a given size and position may depend on the presence of other features of different sizes at related spatial locations.

Multiscale image representations are a powerful tool for analyzing image features at multiple scales. An image is decomposed into a set of descriptions, each making explicit image features at a specific scale. Considerable recent research has focused on the issue of how to best *construct* such a hierarchy of images in coarser and coarser forms. Representations of this type have been referred to as "pyramids", "multiresolution images" and "scale-space images". The total amount of space required to store this type of multiscale description is only a small multiplicative constant greater than the original image.

The second major research question is how to *analyze* and organize the features extracted at many scales with the objective of deriving structural properties of objects in a scene. Of particular importance are the problems of detecting object boundaries and the description of their shape and texture. More specifically, uses of multiscale representations include:

- combination of inter-scale properties to disambiguate possible interpretations of image features (e.g. Marr and Hildreth (1980) hypothesized that edges persisting over several scales are "physically significant"),
- detection of global features by local, parallel procedures at the appropriate coarse scale,
- "coarse-to-fine" search methods which first quickly detect features at a coarse scale and then use these results to localize and verify these and other features at finer scales,
- building a common representation of local and global image features to bridge the "pixel-region gap". A single hierarchical representation can be used in computations involving both local and global features. For example, we may define horizontal and vertical grouping rules based on spatial proximity and property value similarity at different scales.

In this paper we present a unified treatment of some of the main results in understanding multiscale image descriptions. We include work related to the construction (filtering) and analysis of image representations which make explicit properties of object boundary location, shape and texture at multiple scales. This contrasts with other work, often also referred to as pyramid techniques, which does not use multiscale descriptions. For example, other related, but quite different, ideas include (a) pyramid algorithms for rapidly detecting and extracting global image structures by hierarchically grouping and model-fitting image features represented at a single scale in the base level of the pyramid (Rosenfeld, 1986, 1987), (b) quadtrees and other variable resolution structures which represent a single scale image by pixels of variable size (Dyer *et al.*, 1980; Samet, 1984), (c) sequences of functional transformations on single scale images organized into a pyramid-like connection of processes (Uhr, 1972), (d) parameter network hierarchies which compute successively more complex features at each abstraction level (Ballard, 1984; Sabbah, 1985), (e) pyramid architectures (Dyer, 1981; Tanimoto, 1983), and (f) pyramid algorithms for non-image problems (Stout, 1983; Miller and Stout, 1985).

2. Constructing Multiscale Representations

In this section we review work related to the construction of multiscale representations using multiscale filtering techniques. The two predominant representations are (1) *Gaussian pyramids*, which are defined by smoothing brightness values over larger and larger areas producing a set of lowpass filtered copies of the original image and (2) *Laplacian pyramids*, which are defined by differentiating smoothed brightness values producing a set of bandpass filtered copies of the original image. Each level of the Laplacian pyramid makes explicit features tuned to a particular scale and thus forms the basis for most of the work on the analysis of multiscale representations. For example, Laplacian pyramids will be used to detect brightness and texture edges at multiple scales. We summarize some of the major results on these two representations in the following two subsections. The final subsection describes extensions of these two representations to construct color pyramids and texture pyramids.

2.1. Gaussian Pyramids

One of the most common operations performed on images is to blur or smooth the image brightness values. Smoothing enhances an image by reducing the effect of noise so that subsequent processing (e.g. detection of brightness changes) is simplified and regularized. In addition, the amount of smoothing can be adjusted so as to optimally set the resolution at which to locate image features (e.g. edges and texture) which naturally occur at a variety of spatial scales.

Linear smoothing by convolution with a Gaussian-like kernel is equivalent to applying a lowpass filter to the image. The size of the filter is determined by an associated *scale* parameter. In particular, the size of the Gaussian kernel is determined by the scale parameter σ in

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Figure 1 shows a one-dimensional (1D) image that has been smoothed with a set of Gaussian kernels of increasing size. Figure 2 shows a two-dimensional (2D) image which has been similarly smoothed.

As the size of the kernel (scale) increases, the degree of smoothing increases but the bandlimit of the image decreases. Consequently, it is possible to simultaneously reduce the sample rate of the smoothed image in proportion to the bandlimit reduction without loss of information. Using these principles, Burt has shown how to generate a set of Gaussian filtered and sub-sampled copies of an image, each of which has a bandlimit which is one octave lower than its predecessor and the sample rate is one-half its predecessor's sample rate (Burt, 1981, 1983). Because the set of images corresponds to an image convolved with a set of Gaussian filters which double in size and the sample rate is reduced by one-half in each succeeding image, the result is called a *Gaussian Pyramid.*

Each level of the Gaussian pyramid represents a smoothed version of the original image using a different size filter. Notice that the size of the filter is determined by a scale parameter and this determines the sampling Nyquist rate. Although it is not necessary to simultaneously sub-sample the smoothed image at this rate, there are obvious space efficiency advantages of doing so. Consequently, the term "pyramid" is often used synonymously with the terms multiscale and multiresolution. With the octave spacing between levels and associated sub-sampling, the total number of levels in the pyramid is the logarithm of the image size and the total number of image sample points is only 4/3 ($= 1 + 1/4 + (1/4)^2 + \cdots = 1/(1-1/4)$) the number in the original image.

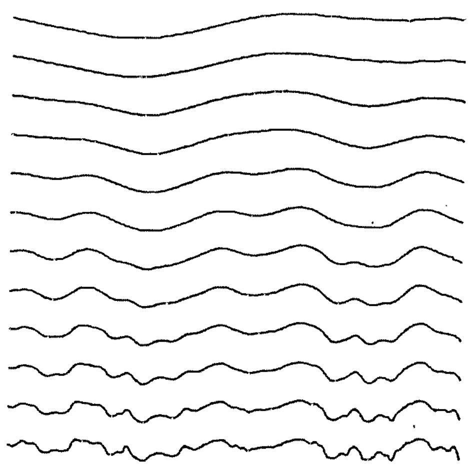

Figure 1. Gaussian smoothed versions of a 1D image.

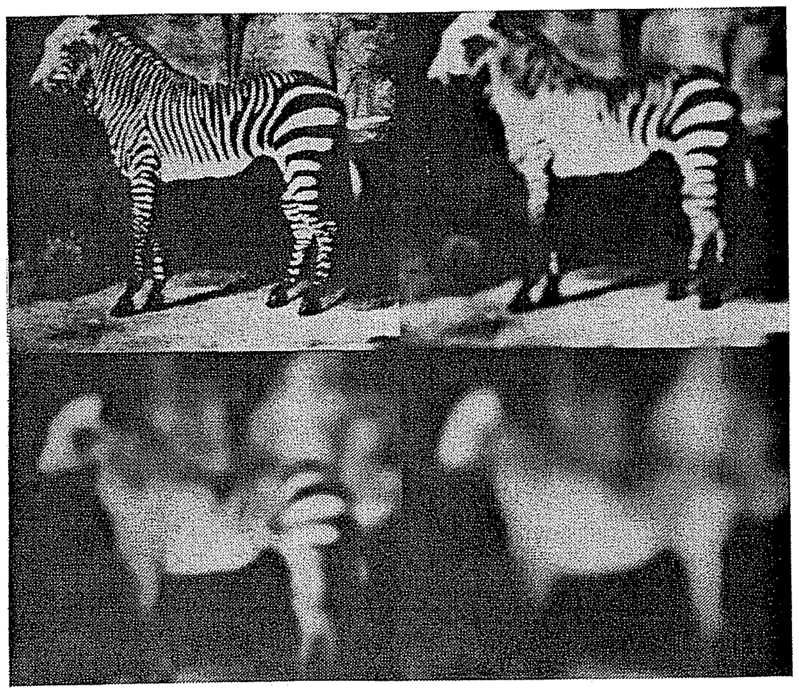

Figure 2. Gaussian smoothed versions of an image of a zebra. Original image is at upper-left. Upper-right corresponds to $\sigma = 2$, lower-left is $\sigma = 4$, and lower-right is $\sigma = 8$.

In general, one can define an arbitrary set of smoothed images using different size kernels and sample reduction rates. For example, Crowley and Stern (1984) use Gaussian filters which increase in size by $\sqrt{2}$ and the sample rate is reduced by $1/\sqrt{2}$. Marr and Hildreth (1980) use Gaussian filters which increase in size by 1.6 and do not reduce the sample rate. Hanson and Riseman's (1980) processing cones allow the user to define the type of smoothing and resampling to be performed. Many others have used unweighted averages over nonoverlapping squares which differ in size by powers of 2; the resolution of each image is also reduced by one-half in each succeeding image (e.g. (Tanimoto and Pavlidis, 1975; Levine, 1978)). The main difficulty with using nonoverlapping averaging windows is that the brightness values at a given level of the pyramid are highly dependent on the position of the original image in the bottom level.

There are several important properties of this type of multiscale lowpass filtered image and, in particular, the Gaussian pyramid. First, each image represents a different degree of smoothing of the original image which reduces noise and textural detail in various amounts. Hence, each image is "tuned" to a particular resolution of image brightness values by blurring all finer resolution features. Second, Gaussian filtering is the only rotationally symmetric operator (for continuous image functions) that is the product of two one-dimensional operators, and it has the smallest product of width in the spatial and frequency domains. Third, Gaussian filtering serves to regularize the image, making subsequent differentiation operations mathematically well-posed (Torre and Poggio, 1986). Fourth, the Gaussian filter has been shown to be the only filter in which zero-crossings of the Laplacian of an image filtered through the Gaussian filter are never created as the size of the filter increases (Babaud *et al.*, 1986; Yuille and Poggio, 1986). Fifth, Gaussian pyramids can be efficiently implemented, for example using Burt's hierarchical discrete correlation method (Burt, 1981, 1983) or Wells' cascaded uniform filters (Wells, 1986). That is, the images are computed level by level by applying a small, fixed size (e.g. 5×5) generating kernel to each level image to obtain the next level image; thus the time complexity of the algorithm is linear in the size of the original image (i.e., is independent of the effective kernel sizes).

2.2. Laplacian Pyramids and Edge Pyramids

Just as the degree of smoothing is varied in producing the Gaussian pyramid, spatial changes in brightness can be detected over various size regions to generate a set of bandpass filtered copies of the original image. This is important for detecting edges at various scales. The biological motivation for multiscale edge detection is extensive (e.g., see (Wilson and Giese, 1977; Wilson and Bergen, 1979)). Although various differentiation operators could be used for this purpose, the Laplacian operator, ∇^2, is the most common. Combined with the Gaussian smoothing function, we obtain the operator

$$\nabla^2 G(x, y) = \frac{1}{\pi\sigma^4}\left[\frac{x^2+y^2}{2\sigma^2} - 1\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

We will use the term *Laplacian pyramid*, originally coined by Burt and Adelson (1983), to describe this type of multiscale representation. Other related representations are Crowley's difference of lowpass transform space (Crowley and Parker, 1984), Witkin's scale-space images (Witkin, 1983), edge pyramids (Levine, 1978) and zero-crossing pyramids. We will use the terms *zero-crossing pyramid* and *edge pyramid* to describe the positions where edges have been detected in an image, for example as determined by where the sign changes in each level of the Laplacian pyramid. Figure 3 shows the output of the $\nabla^2 G$ operator applied to a 1D image. Figure 4 shows two levels of the zero-crossing pyramid for a 2D image of a zebra.

The computation of the Laplacian pyramid can be done in several ways. The Laplacian operator can be applied to each level of the Gaussian pyramid to obtain the corresponding level in the Laplacian pyramid. Alternatively, each level can be computed directly from the original

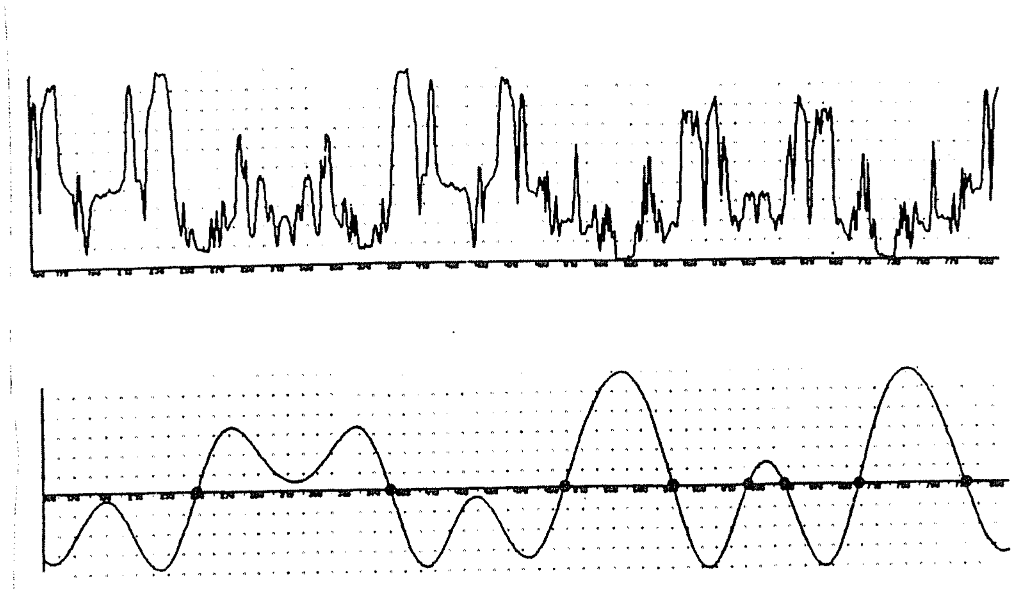

Figure 3. Result of applying a $\nabla^2 G$ operator with large σ to the 1D image at top.

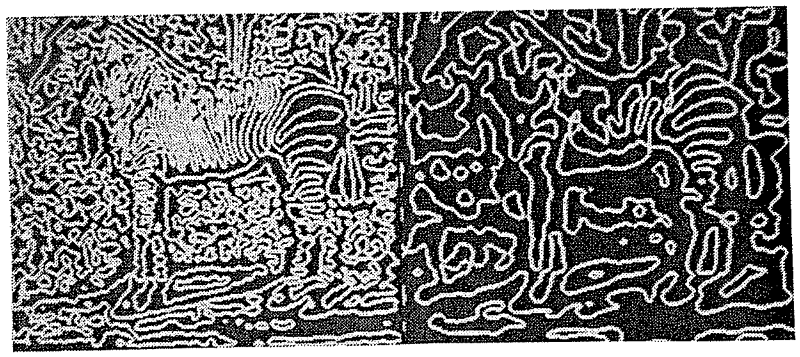

Figure 4. Two levels of the zero-crossing pyramid for the zebra image. Left corresponds to $\sigma = 2$ and right is $\sigma = 4$.

image using the $\nabla^2 G$ operator with a particular scale parameter, σ. Finally, the $\nabla^2 G$ operator can be approximated by a difference of two Gaussian functions of the appropriate sizes; this operation can be performed very efficiently as the difference of two adjacent levels of the Gaussian pyramid (Burt, 1981, 1984; Burt and Adelson, 1983; Crowley and Stern, 1984).

The motivation for Laplacian pyramids is the assumption that for a given image there exists one or more scales such that differentiation of the smoothed brightness values at each of the given scales will transform the image into a set of representations in which each physical edge of interest in the scene is more explicitly marked. Thus the boundary between pairs of adjacent regions will be observable by differentiation of an appropriately smoothed image which blurs the texture inside each region.

For example, in a very high resolution aerial image of a forest, rapid local fluctuations in brightness could be caused by the edges of individual leaves. To delineate individual trees, one could first smooth the image intensities using an appropriate lowpass filter and then detect brightness changes in this medium resolution image. To separate adjacent stands of different tree types, one could use an even larger smoothing kernel before differentiation.

The zero-crossings in each level of a Laplacian pyramid indicate the positions of edges at each scale. Multiscale edge detection has been used for two purposes: (1) computational efficiency in detecting edges in the finest resolution image and (2) the computation of edges at multiple scales in order to better analyze the underlying physical causes of the brightness changes. The first approach originated with the work of Kelly (1971) on a two scale, coarse-to-fine "planning" edge detector which used the results from the coarse scale to more selectively apply the edge detector at the fine scale only where needed. This sequential approach saves needless work in uninteresting areas of the highest resolution image and also serves to verify and localize the results obtained at the coarse resolution. Numerous extensions of this approach have been defined and are discussed in Section 3.1.1.

The second purpose for multiscale edge detection is to use information from a set of edge detectors tuned to different scales to better analyze the physical properties in the given scene. Consequently, an important aspect of this problem is to build a description of edges such that each edge can be tracked across multiple scales so that corresponding brightness changes can be compared and analyzed. Probably the earliest approach to analyzing edges across multiple scales was Rosenfeld and Thurston's (1971) method for selecting the "best" edge scale from the outputs of a set of first derivative operators defined by computing differences-of-averages over blocks which increase in size by powers of 2. Similarly, Marr and Hildreth analyzed zero-crossings of the Laplacian of a Gaussian smoothed image over several scales in order to locate edges which were "physically significant" (Marr and Hildreth, 1980; Marr, 1982). Hartley (1985) described an implementation of an edge pyramid based on the Hueckel edge operator.

Both the work of Rosenfeld and Thurston, and Marr and Hildreth are image-centered, i.e., an edge detector is applied across all scales (kernel sizes) centered at each pixel in the original image. Alternatively, one could first detect edges at each scale and then "track" a given edge continuously over a range of scales, defining a more object-centered approach. This is the basis of the "scale-space" approach developed by Witkin (1983) and described in Section 3.2.1.1. Recently, a number of "nice" inter-scale properties of edges have been proved which make this "tracking" process feasible. First, zero-crossings of the Laplacian of a Gaussian filtered image are not created as the scale of Gaussian smoothing increases and the Gaussian filter is the only filter that has this property (Yuille and Poggio, 1984, 1986; Babaud *et al.*, 1986). Second, the complete set of zero-crossings across all scales uniquely characterizes the original image (up to a constant) and hence defines a complete representation (Burt, 1981; Yuille and Poggio, 1984, 1986).

2.3. Color Pyramids and Texture Pyramids

The ideas used in generating Gaussian (lowpass filtered), Laplacian (bandpass filtered) and zero-crossing pyramids based on brightness values can easily be extended to other types of point and local property values. For example, color images can be used to define the bottom levels of a set of pyramids, one for each of the primary colors red, green and blue. *Color pyramids* have been used by several researchers (Hanson and Riseman, 1978; Levine, 1978, 1980). In this case the Gaussian pyramid represents blurred versions of intensity of a particular color. The Laplacian pyramid represents multiscale color edges. Alternatively, color pyramids can be defined based on other color space systems including hue-saturation-brightness and opponent color systems. More generally, a pyramid could be constructed for each spectral band in a multispectral image.

The Gaussian pyramid can also be used to blur local properties of brightness values for texture feature measurement, e.g. edge density, local power spectrum, texture energy measures, and cross-correlation with a template. Gaussian pyramids defined by blurring some local property value will be referred to as *texture pyramids.* Various types of texture pyramids have been defined (Levine, 1978, 1980; Burt, 1983, 1984; Kjell and Dyer, 1986, 1987). In Section 5 we describe a method which uses such a texture pyramid for image segmentation. Other intrinsic image features such as disparity, local surface orientation, and optical flow could also be used as the basis for constructing a Gaussian pyramid.

Laplacian and zero-crossing pyramids can also be defined to detect multiscale edges of an arbitrary local property value. For example, there are many types of texture regions which are visibly distinct but have identical mean brightness values. Therefore, differentiation of some local property value other than brightness, e.g., local brightness variance, is necessary in order to locate the boundary between these two texture regions.

3. Analyzing Multiscale Representations

The analysis of properties detected at multiple scales is a very difficult problem that is only just beginning to be understood. As mentioned in the Introduction, there are a number of ways that researchers have used multiscale properties for image understanding. This includes (a) using the persistence of a property over a range of scales as a heuristic for inferring important image features, (b) "coarse-to-fine" search for a given feature, (c) detection of global structure using local operations at coarse scales, and (d) hierarchical perceptual organization to group similar objects (detected at various positions and scales) together into global structures. The common aspect of all these applications is that each defines a procedure for extracting important structural properties by relating the descriptions at multiple scales to one another and coordinating their use. In this section we review some of these methods and then present two recent pyramid algorithms developed by the author for model-based object recognition (Section 4) and texture segmentation (Section 5).

3.1. Coarse-to-Fine Operations

When properties are known *a priori* to (1) vary smoothly with scale and (2) spatially coincide across all scales, then it can be a significant computational saving to design "coarse-to-fine" operations. These procedures first apply operators to coarse (large smoothing kernel) images and then use the results to either focus attention or refine results in fine (small smoothing kernel) images. These methods do not attempt to combine the descriptions at different scales, except in the sense of constraining the computation at the next finer scale. The following sections illustrate some of the uses of this idea for detecting features by a sequence of local operators at successively finer scales and region interpolation of property values using approximate results from a coarser scale.

3.1.1. Feature Detection

The most direct use of a multiscale representation is for efficiently detecting large image features by applying a small operator to a coarse scale (small) image rather than a large operator to a fine scale (large) image. This converts global features into more local ones. Unfortunately, very little theoretical work has been done which specifies exactly how certain geometric features are distorted in Gaussian, Laplacian, or zero-crossing pyramids. Tanimoto (1976) analyzed the effects of smoothing in nonoverlapping, averaging pyramids on features such as spots and edges. In general, operators which detect large features by matching small, coarse features with coarse resolution images are not completely reliable; therefore results can only be used to indicate a "fuzzy" confidence of match.

Coarse-to-fine procedures solve this problem by sequentially applying operators to successively finer scale images, at each level refining and verifying the matches made at the previous level. These methods have three main advantages: (1) they iteratively "zoom in" on the exact position of a feature represented at the finest resolution, (2) they verify an hypothesized feature tentatively detected at a coarser scale and (3) they provide a computationally efficient means for applying expensive matching operators only in areas of interest. The main disadvantage is that features may be missed at a coarse level even though they are present in the finest level. Most implemented procedures have used just two levels and correlation-based matching techniques.

The most direct application of this coarse-to-fine detection technique is to "track" a single feature from coarse to fine; this includes edge and boundary detection (Kelly, 1971; Tanimoto and Pavlidis, 1975; Baugher and Rosenfeld, 1985; Hartley, 1985a; Borgefors, 1986) and general template matching (Rosenfeld and VanderBrug, 1977; Wong and Hall, 1978; Tanimoto, 1981). In other problems, features must be matched across two or more images and this approach can be used to successively shrink the search area for corresponding features by first matching a pair of coarse images and then using the results to localize the matching at the next finer level;

applications include stereo correspondence (Mori *et al.*, 1973; Moravec, 1977, 1980; Marr and Poggio, 1979; Grimson, 1981; Liu *et al.*, 1986), motion analysis (Burt, 1984; Anderson *et al.*, 1985), and image registration (Glazer *et al.*, 1983). Parallel implementations of this technique for stereo and motion algorithms on a "mesh-connected" array computer are given in (Williams and Anandan, 1986). It has been shown (Clark and Lawrence, 1986), however, that errors in the location of features increase with coarseness, and therefore the comparison and matching of features in multiple images can result in errors at the coarse levels of matching. Finally, the detection of a given feature at one scale can be used to guide the search for a different feature at a different scale, but at a known spatial relation to the first feature (Neveu *et al.*, 1986). In Section 4 we describe a procedure that we developed which uses this idea to model objects in terms of their parts, each represented at its most appropriate scale.

The speedup in using a coarse-to-fine algorithm instead of the single scale algorithm is significant. Given an $n \times n$ image and a feature specified by a $k \times k$ template, the brute-force algorithm requires $O(n^2k^2)$ arithmetic operations. Using a scale factor of s, we can instead apply a $\frac{k}{s} \times \frac{k}{s}$ template to a coarser image of size $\frac{n}{s} \times \frac{n}{s}$. This results in $O(n^2k^2/s^4)$ operations at the coarse scale. For each tentative match, we must verify and localize the feature at the fine scale by testing the $k \times k$ template at each point in the $s \times s$ block corresponding to the coarse scale match position. This requires a total of $O(s^2k^2)$ operations. Thus the total number of operations is $O(n^2k^2/s^4 + ps^2k^2)$, where p is the number of tentative matches found at the coarse scale. The speedup for the two-level procedure is $O(s^4/(1 + ps^6/n^2))$; since p is usually a small constant and $s \ll n$, this results in an improvement of approximately s^4.

Extending this method to more than two levels further improves the speedup as follows. Consider a set of $m+1$ successively coarser scale images of size $n \times n, \frac{n}{2} \times \frac{n}{2}, \frac{n}{4} \times \frac{n}{4}, \ldots, \frac{n}{2^m} \times \frac{n}{2^m}$. The analysis in the previous paragraph implies that at the coarsest level $(m+1)$ the number of operations performed is $O(n^2k^2/2^{4m})$. For each match at level $m+1$ we must perform $O(4k^2)$ operations at level m because the scale change between these two levels is 2. Similarly, in order to move from any level to the next finer level, $O(4k^2)$ operations must be performed. Assuming p features are originally detected at level $m+1$ and all of them are tracked down to the finest resolution image, the total number of operations is $O(n^2k^2/2^{4m} + 4pmk^2)$. Using just two levels with $s = 2^m$ would require the same number of operations at the coarsest level followed by $O(p\,2^{2m}k^2)$ operations at the fine level. Thus by tracking the features at m successively finer levels rather than just 1, there is a speedup of $4pmk^2/p\,2^{2m}k^2 = m/4^{m-1}$ operations.

Finally, the success of this strategy depends on the successful detection of features at each level. In general, the distortion introduced due to the blurring in the Gaussian pyramid and image noise will make the decisions at each level difficult. In practice, a predefined cost function is used to evaluate each match and a threshold is used to reject matches which are clearly poor. Of course, the specification of the "best" match position as we track a single feature from coarse to fine is feature-specific and therefore little theoretical work has been done on the properties of these cost functions.

3.1.2. Region Interpolation

Multiscale region interpolation methods use constraints both within each level and between adjacent levels to efficiently compute the values in the finest resolution image. A smoothness assumption is used as the primary constraint between levels.

In many cases we are given complete information about an image (region) at one level and the goal is to reconstruct the image (region) at the next finer level. Additional constraints about the new image may or may not be specified (e.g., a sparse set of points may have known values). The simplest problem of this form is specified when the value of a point in the new image is only dependent on a local neighborhood of point values in the coarse image. This is the inverse of the Gaussian pyramid construction problem. Interpolation techniques of this form are often referred to as "projection" (Hanson and Riseman, 1978) or "expand" operations (Burt, 1984). Various types of interpolation procedures have been used depending on the size of the neighborhood of points in the coarse image used to compute each point in the new image and the weight associated with each point in the neighborhood.

More generally, the value of a point in the new image may depend globally on the values of all other points in the image. Global optimization problems of this type which can be formulated in terms of local constraints can often be solved by iterative relaxation methods. Conventionally, these methods are applied to an image at a single resolution. The new pixel value at each iteration is dependent on the previous values of the pixel and its neighbors. To decrease the number of iterations required to achieve global convergence, multigrid relaxation can be used to iterate at multiple scales. For example, Narayanan *et al.* (1983) first iterated on a coarse image to derive an approximate solution, then interpolated this solution to obtain an initial estimate of the fine image values and finally iterated on the fine image to obtain a final solution. Alternatively, all levels can be iterated simultaneously, each level cooperating with its adjacent levels until the finest resolution level is solved. Terzopoulos (1983, 1984, 1986) studied this approach for visual surface reconstruction, recovery of lightness, shape-from shading and optic flow. Glazer (1984) used this approach for optic flow computation.

3.2. Property Organization Operations

One of the main issues associated with multiscale representations is how to infer significant scene structure from the relations between image properties measured across multiple scales. Coarse-to-fine operations assume a simple, known correspondence between properties at multiple scales (usually based on spatial coincidence), and do not attempt to build a composite description containing features at various scales and spatial locations. In contrast, in the methods described in this section, the goal is to recover structure by analyzing combinations of features occurring at multiple positions and scales.

Perhaps the most straightforward method of this type is to compute a vector of property values at each pixel, corresponding to the response of a given operator across a range of scales. Each vector can then be analyzed to select the "best" scale and associated value at a given point. This type of analysis is useful with images in which there is no *a priori* knowledge about which scale is appropriate for detecting a given object or property. Spatial coincidence of property values must be assumed, however, so that the outputs of the operators applied at a given point at each scale correspond to the same physical cause.

For example, Rosenfeld and Thurston (1971) used the outputs of edge detectors of various sizes to select the most "conspicuous" edges. That is, at each image point the "best" scale is selected based on the vector of outputs centered at the point. Marr and Hildreth (1980) developed a similar idea and applied it to the analysis of a set of zero-crossing images at multiple scales. An edge is detected and verified as "physically significant" if zero-crossings occur at a given point over a sufficiently large, contiguous range of scales.

More generally, vectors could be computed using arbitrary local property operators of various sizes. For example, each vector could represent a description of the texture centered at a point using larger and larger window sizes, and this could then be used for texture analysis (Peleg *et al.*, 1984; Werman and Peleg, 1985).

More generally, boundary and/or region features can be detected independently at each scale and then "linked" together, defining explicit "tracks" or "contours" of each feature in scale-space. The following two sections consider these two types of features separately.

3.2.1. Curve Analysis

In this section we consider the analysis of multiscale curvilinear data representing region boundaries or curves or 1D images. There are two alternative approaches to the problem of analyzing region boundaries at multiple scales: (a) first extract the boundary at the finest scale and then construct and analyze the multiscale description of this curve, or (b) first construct the multiscale description of the 2D image, extract the boundary at each scale and then link the boundary points across scales. The first case is simpler because the boundary can be parameterized by a one-dimensional function of curvature as a function of arc length along the boundary. Thus the intrinsic form of the boundary is smoothed at multiple scales. The second case represents curves as the boundaries of regions and the image containing the regions is smoothed rather than the region boundary or curve itself. Consequently, this case must consider scale-space "surfaces" instead of contours in tracking features across scale.

Smoothing the boundary directly across multiple scales is clearly advantageous when the boundary can be reliably extracted from an image. In general, however, the presence of multiple touching or overlapping objects and noise can make the process of extracting complete boundaries very difficult.

Below we consider these two cases separately as 1D and 2D multiscale curve analysis.

3.2.1.1. One-Dimensional Curves

In this section we consider the case of analyzing either 1D images given by the image function $z = f(x)$ representing brightness z as a function of spatial location x or 1D curves given by the parametric equation $k = K(t)$ specifying the curvature k as a function of the arc length t for a 2D object boundary starting from some arbitrary boundary point. Given a multiscale description of such an image or curve, the goal is to explicitly relate features detected at various positions and scales. Of course, the interpretation of these features is quite different in these two cases since one represents smoothed image features and the other represents smoothed boundary features. Applications include 1D images, hyperspectral descriptions of a single pixel, and boundary or axial curves extracted for an object in a 2D image.

Witkin (1983) introduced the *scale-space representation* which provides an explicit description of the persistence of edges over scale. A scale-space image is a description of the zero-crossings detected at multiple scales in a 1D image. Zero-crossings resulting from the same edge in the scene are then linked together forming contours which are either lines from small to large scale, or bowl-like shapes as shown in Figure 5. The bowl-like contours are closed at the top (large scale) and open at the bottom. We will use the terms *scale-space contour* or just *contour* to refer to this type of linked multiscale description of a single feature, such as an edge. The geometric constraints of these scale-space contours are such that rarely do contours intersect each other. Therefore each bowl-shaped contour is strictly related to all other contours; i.e., a contour is either to the left of, contained in, or to the right of the given contour. This means that all contours can be organized hierarchically into a 3-ary tree where each node stores a contour and its subcontours are stored in the appropriate one of its three subtrees. Witkin defined a particular version of this *scale-space contour tree*, called an interval tree, which was used to select the most "stable" physical edges corresponding to the contours that persist over a wide range of scales without interference from other nearby containing contours in the tree.

An important property of the scale-space representation is that it makes explicit the regions in a 1D image determined by pairs of bounding zero-crossings. That is, a bowl-like contour exists for each 1D region and these regions are organized hierarchically as determined by the strict nesting relationship that must exist between contours. The interval tree represents a symbolic description of this organization.

Asada and Brady (1986) have expanded on Witkin's approach by searching for specific primitive features in scale-space. Rather than use 1D images, however, they use the 1D

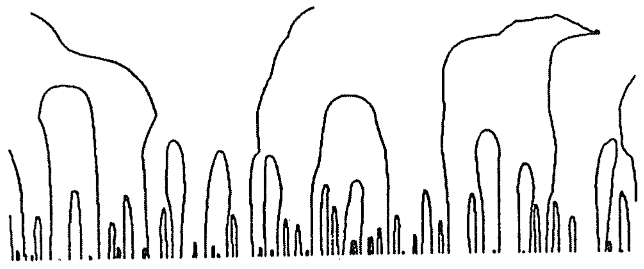

Figure 5. Scale-space representation of the positions of zero-crossings over a continuous range of values of σ.

parametric function (parameterized by the curve's tangent direction as a function of arc length) of the boundary of an object previously extracted from a (fine-scale) 2D image. A pair of Laplacian-like pyramids are constructed using first and second derivatives of the Gaussian pyramid. Next, local positive maxima and local negative minima are linked across scales into scale-space contours and then organized into a contour tree. Finally, the tree is searched for specific patterns of contours which imply the presence of "interesting" curve primitives such as corners, joins, ends, cranks and bumps. Contour patterns are defined for each primitive type by deriving a set of multiscale constraints which in turn imply a set of relations on the types and positions of contours in scale-space. For example, corners are recognizable in scale-space by a pair of contours from the second derivative pyramid which converge linearly with scale and are centered on the given corner. Other primitives have other distinct contour patterns in scale-space. The analysis of scale-space involves finding the best matching primitives to account for most of the contours in scale-space.

Mokhtarian and Mackworth (1986) have defined a matching technique for scale-space contours and applied it to the problem of registering an image with a map. Given a 2D image of an object, its boundary is extracted and parameterized as curvature as a function of arc length along the boundary. The zero-crossing pyramid is then constructed, followed by the associated scale-space contour tree, where each contour corresponds to a boundary inflection point in scale-space. The matching procedure is a uniform-cost search which compares contours top-down through the two given contour trees. The cost of matching a pair of contours is given in terms of two parameters which transform one contour into the other one.

3.2.1.2. Two-Dimensional Curves

In this section we consider the analysis of multiscale curves represented in the image plane as either boundary or axial points of regions. Region boundary and axial curves can be represented over multiple scales since a 2D region is "visible" in the Gaussian pyramid over multiple scales. That is, each level of the Gaussian pyramid effectively "shrinks" a region by blurring boundary points sufficiently so that they become part of the background and are hence "removed" from the region. Suppose each level shrinks the region by removing all border points. (In general, the number of points removed depends on the gray levels of points in the region and background, and on the width of the region in the neighborhood of each border point.) We define the *thickness* of the region as the number of shrinking steps or levels required to completely delete the region. The curves that we analyze in this section consist of points detected at all levels up to the level associated with the thickness of the associated region.

The main difficulty in trying to generalize the results of the last section to 2D is that the shape of a region varies locally in both boundary curvature and region thickness. Consequently, the Gaussian pyramid representation of the region can become disconnected at coarse scales and the boundary may be a complicated sequence of variable scale features. Thus, in general, the 2D analog of a region's scale-space contour is not a simple unimodal, bowl-shaped contour.

We assume that curves are specified at each resolution by the characteristic function $z = f(x, y)$, where $f(x, y) = 1$ if the given pixel is part of a curve and 0 otherwise. For example, given the zero-crossing pyramid, which specifies at each scale those points that are part of the boundary of an object, we would like to define 2D counterparts to the 1D scale-space contours and contour trees that were defined in the last section.

Note that we do not include the analysis of images containing curves because these objects have thickness 1 and are therefore only visible at a single scale. Related, but non-multiscale, pyramid methods for analyzing curves hierarchically by coarser and coarser approximations have been studied (Shneier, 1981; Hong *et al.*, 1983; Hartley and Rosenfeld, 1985; Kropatsch, 1985).

Crowley defined a multiscale description similar to Witkin's based on the explicit linking of information in a Laplacian pyramid called the DOLP space (Crowley and Parker, 1984; Crowley, 1984). Lacking a generalization of the type of nice geometric properties of zero-crossing contours for 1D images used by Witkin, Crowley used a more *ad hoc* set of rules for linking points across scale. Instead of detecting zero-crossings to mark the edge points, however, Crowley detects local maxima and minima in both one (ridges) and two (peaks) dimensions at each level producing a pyramid of axial points (also called skeletal, medial-axis and symmetric-axis points). These points are linked together both within each level and across adjacent levels to form a set of scale-space contour trees. Points which are peaks and also local maxima across adjacent scales, are marked as special "landmark" points in each tree. These points are considered to be the perceptually most salient features and are used to represent the key parts of an object which are used for coarse matching.

A number of investigators have considered the problem of linking region boundary points across scale (Hong *et al.*, 1982b; Hong and Shneier, 1982, 1984; Fleck, 1986). Initially, a representation of the boundary points is constructed such as the zero-crossing pyramid. Next, rules based on proximity and good continuation are used to locally link points within each level and between adjacent levels. The resulting structure is analogous to the 1D scale-space contour tree, although in 2D a region may be represented by a set of contour trees corresponding to parts of the region which become disconnected at coarse scales. Unfortunately, no theoretical analysis has been done of the properties of these 2D contour trees as a function of region shape.

Given a set of 2D contour trees resulting from the linking of boundary or axial points, various global region properties can be computed by local analysis of the contour tree. For example, compact regions of known size can be detected by searching for an antiparallel pair of edge points at the appropriate level of the contour trees (Hong *et al.*, 1982b). Similarly, various axial shape properties can be computed from these contours (Fleck, 1986). Little work has been done, however, on computing global shape properties by combining features from multiple levels of contour trees.

3.2.2. Region Analysis

The last section considered the problem of linking boundary or axial points into contour tree descriptions of a region. Rather than build a surface description of region boundaries or axes in scale-space, one can alternatively build a *volumetric* description of region interior points in scale-space. Thus the scale-space contour tree corresponds to a representation of the volume of scale-space filled by a given 2D region.

The basic procedure for building such a description is to merge points into the same tree based on properties of similarity and proximity. That is, each region is assumed to be homogeneous with respect to some local property such as brightness, color, or texture. A Gaussian pyramid is constructed based on this characteristic property and then points are linked within each level and across adjacent levels based on their proximity and similarity of property values. Burt *et al.* (1981) were the first to develop rules for linking points into multiscale region descriptions.

The main difficulty with this approach is the "window selection problem". That is, region properties are computed over a local window of pixels whose size is determined by the textural properties of the region. For example, if a region consists of a microtexture, a small window should be used; macrotexture regions are best described by properties computed over windows of size comparable to the texture pattern size. Of course, given an image we do not know *a priori* how the image should be segmented into areas of varying texture size. Therefore we cannot know what is the best window size to use at each position in the image. Furthermore, near the boundaries between regions the window should be chosen so that it does not overlap the boundary and includes only pixels which are contained in a single texture region; this implies that the shape of the window should also vary from point to point. Because of this problem, Burt argued that two general computational strategies are needed: (1) properties should be computed at multiple scales at all positions in an image so that all possible window sizes are used, and (2) segmentation and image properties should be iteratively computed in a cooperative fashion so that the tentative results from each task are used to adjust and improve the performance of the other task at each iteration. The second requirement allows the window shape to be adjusted locally based on the current best estimate of how any region boundaries may pass through the given window.

More specifically, rather than compute the Gaussian pyramid using a fixed size window at each level, each point's blurred property value is computed by averaging a subset of the points in its candidate window at the next finer level. The selection of the subset of points is determined by the current best segmentation. A segmentation of the image at a given level is achieved by having each pixel select its "most similar parent" pixel from among a set of candidate pixels at the next coarser level. Similarity is defined by the absolute difference in property values of two given pixels. The set of candidate parent pixels includes all pixels at the next coarser level which spatially overlap the given pixel. Thus each pyramid pixel's updated property value is computed as the average of all of its candidate children pixel's which are currently "linked" to it as their best parent. The segmentation is updated by having each pixel recompute its best parent. By alternating these two processes, coarse level pixels should eventually contain a property value which is the average of the property value of pixels contained in a single region in the finest level image. A number of variations on this approach have been investigated including weighted instead of binary linking rules (Hong and Rosenfeld, 1982, 1984; Hong *et al.*, 1982a) and alternative ordering strategies for relinking (Silberberg *et al.*, 1981; Cibulskis and Dyer, 1984a, 1984b).

4. Multiscale Model-based Object Recognition

Most of the techniques developed to date for model-based object recognition use a set of features computed at a single scale (e.g., (Bolles and Cain, 1982)). In this section we describe a system which combines multiscale object modeling with coarse-to-fine matching. Multiscale object modeling is important because in general no one scale is best for describing all aspects of an object to be recognized. For example, a model of a "key" may describe the "shaft", "head", "hole" and "teeth". The shaft and head may best be described at a relatively coarse level in which the shaft is described as a ribbon and the head as a circle. The hole may be appropriately defined at an intermediate scale and the teeth at a fine scale. Trying to create a model which includes all of these features at a single scale would not only be less natural, but would also require more storage space since the finest scale would be used for describing all the features.

To avoid these problems, we will model two-dimensional objects hierarchically using a directed acyclic graph. Each node in the model graph describes a user-selected, not necessarily connected, segment of the object boundary at one scale represented by one pyramid level. Arcs are directed from boundary features at one scale to boundary features at finer scales.

The hierarchical model graph is used to guide a coarse-to-fine matching procedure. Briefly, the coarsest scale boundary description of the object is stored as the root node of the model graph and is matched first with the corresponding level of an input image's zero-crossing pyramid. The results of this match are used to hypothesize a list of approximate positions and orientations of the object. Next, successor nodes in the model graph are matched with sub-windows of the larger, finer scale zero-crossing images to verify the presence of finer boundary features and to determine more precisely their positions and orientations. Thus the model graph is an explicit representation of the focus-of-attention strategy for the given object. The sub-windows are chosen according to the rough estimates provided by the coarse level match positions. The generalized Hough transform is used for all the matching processes (Ballard, 1981).

In the remainder of this section we present the details of the algorithm. Section 4.1 details the method of constructing the model graph. Section 4.2 presents the hierarchical Hough matching algorithm using a zero-crossing pyramid and model graph. Pilot results are presented in Section 4.3.

4.1. Model Construction

One drawback of many object recognition algorithms is that objects are represented by connected "chains" of boundary pixels such as ribbons, skeletons, or concurves. These representations are intuitively appealing and may require less memory than the zero-crossing image from which they were derived, but they are time consuming to generate because of the serial operations of edge point linking and curve fitting.

In order to avoid this serial processing bottleneck, we represent objects as simple sets of unconnected edge points in an image window. The features stored in the model graph are vectors (stored in a generalized Hough transform R-table) corresponding to the edge points in selected windows at specified levels of a zero-crossing pyramid. Figure 6 shows four levels of the zero-crossing pyramid derived from an image of some keys, a washer and a screw.

Given a set of zero-crossing pyramids, one for each prototype object to be modeled, we next construct a *models feature graph,* i.e., a graph of local features in which subgraphs specify individual object models. For simplicity of exposition, we assume there is only a single object being modeled. The merging of object graphs into a single composite models feature graph is straightforward.

Currently, the user interactively selects at each level of the zero-crossing pyramid a set of windows, each containing edge features of interest for the object being modeled. From each window, a node in a directed acyclic graph is constructed. Each node stores (1) the level of the pyramid from which the window was taken, (2) the edge points in the window (stored in an R-

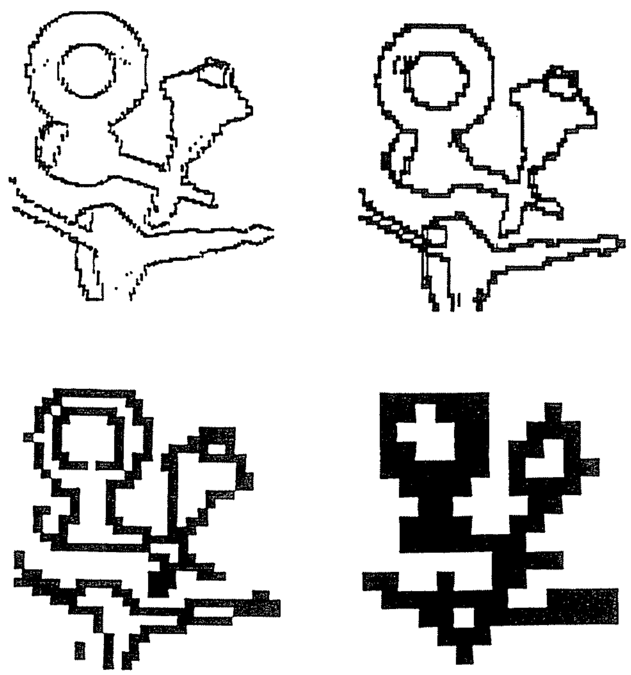

Figure 6. A four level zero-crossing pyramid.

table (Ballard, 1981)), (3) pointers to children nodes, if any, (4) a weight used to indicate the significance of the feature contained in the window and (5) the window's position and orientation relative to the centroid and major axis of the complete object (computed from the finest scale zero-crossing image). The root node of the graph always contains the entire coarsest scale zero-crossing image (currently 16×16) of the prototype object. Each node in the graph may have zero or more children, with the features stored in each child sometimes, but not always, contained in a finer scale sub-window of its parent's window. Beginning at the root node and moving from parent to child, the features stored at each node must be a finer scale description of part of the object than its parent node's description.

It is not necessary that a child node contain features from the next finer scale image of the object, i.e., it is possible to skip levels between adjacent nodes in the model graph. This is not recommended, however, for the following reason. The hierarchical Hough matching algorithm described in the next section uses the maximum peak in the accumulator array for the current node to estimate the location of the window where the child node is expected to be found. This estimate is based on the estimated location of the object's centroid with respect to the current node and is always integral, with accuracy of plus or minus one-half pixel. Therefore, since the sample rate of each pyramid level is twice that of the previous level, the location of the window where the child node should be found is at best correct to within one pixel. If the peak location in the parent's accumulator array is chosen incorrectly by one pixel, then the location of the child's window will be off by two pixels, plus or minus one, or a total of up to three pixels. If the child node were associated with a window which is sampled at a rate higher than twice its parent's sample rate, then these errors are compounded even further. Large errors in window position will often cause the matching algorithm to fail because the error size becomes a significant percentage of the size of the window itself.

The weight associated with each node indicates the significance of this feature for the overall recognition of the object. Static weights are chosen by the user so that during the matching process if there exists any connected subgraph of nodes with total weight greater than 1, then sufficient evidence exists to indicate the detection of the object. Thus each such *match-set* of nodes contains a sufficient number of features to uniquely distinguish the object from other possible objects. Specification of match-sets is important not only to speed the recognition process, but also to correctly detect objects which are partially occluded or noisy.

For example, if the object we are modeling is a room key and the other objects of interest are nuts and washers, one of the key's match-sets need only contain the root node and a coarse representation of the shaft of the key. However, if the other objects are different types of keys and are distinguishable only by the patterns of their teeth, then the match-sets need to contain enough fine detail of the teeth in order to discriminate this key from all other types. Currently, all match-sets must include the root node due to restrictions in the matching algorithm. This condition could easily be removed, however, in order to deal with images in which occlusion prevents matching the root node with the coarsest scale image.

The nodes in a match-set usually contain enough fine subfeatures so that the fine scale nodes can verify the matches of their coarser scale ancestors. Sometimes, however, match-sets include nodes that are not subfeatures of their parent nodes. These nodes usually have two or more parents and contain a subfeature of one of the parents but not of the other, so that if one parent is occluded but the other is not, the node can still be reached through the non-occluded node. This is why in the model graph shown in Figure 7 each node in the coarsest three levels is connected by an arc from all possible nodes at the next coarser level.

Figure 7 shows the multiscale model graph for a "key". The model was constructed from a zero-crossing pyramid of the key containing four levels — 16×16 through 128×128. In general, the choice of which image scales and which windows (boundary features) should be used is very application specific. For a given set of model objects, the finest scale should include enough detail to reliably distinguish between each object type. The coarsest scale should be chosen so that each object is visible and contains enough edge points to reliably rank the best match positions at this level. Also, it is important to choose coarse level nodes which can quickly disprove false coarser level matches.

In the model in Figure 7, all nodes' windows are 16×16. Node 1 is the head and node 2 is the shaft of the key from the 32×32 zero-crossing image. Node 3 corresponds to the hole at the top of the head of the key and node 4 is a finer scale view of the shaft. Note that in this case both new nodes have been linked from both nodes at the 32×32 level. This was done for the reason mentioned above, namely, to define match-sets which skip certain coarse scale nodes (i.e., the nodes at the 32×32 level) in case occlusion or noise prevents a successful match at that level. Nodes 5, 6 and 7 correspond to one side of key head, the tip of the shaft and a section of teeth, respectively.

The weights assigned to each node in the model graph are also shown in Figure 7. The root node has been assigned a weight of 0 because we assume in the current implementation that it must be part of every match-set and must always be successfully matched in order to recognize the key. This restriction is not crucial, however, and can easily be removed with appropriate modifications in the matching algorithm. With this assignment of weights there are eight match-sets implicitly defined. Figure 8 shows several of these subgraphs.

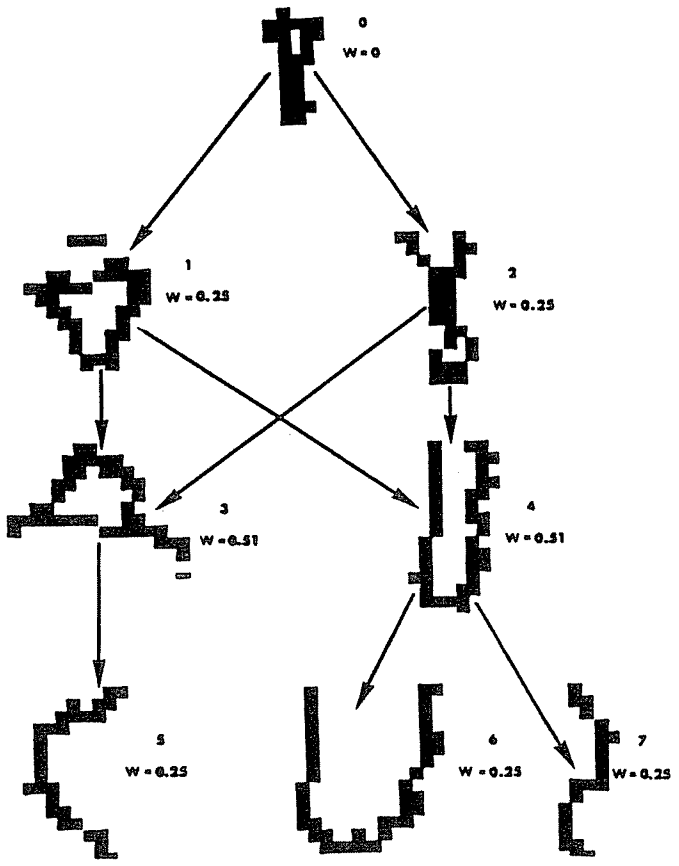

Figure 7. The model graph of a room key.

4.2. Matching

Our matching algorithm employs a top-down, coarse-to-fine search strategy to recognize objects in a given input image. We consider here the particular problem of detecting one instance of a given object specified by its model graph. Our method may be briefly summarized as follows. The initial best estimate of the position and orientation of the object is determined from information at the coarsest scale, and this estimate controls the search for finer scale features. If enough finer scale features are found to uniquely identify the object, the match succeeds. If enough finer scale features are not found, the previous estimate is discarded and the next best estimate at the coarsest level is chosen and used to control the search for sufficient confirming finer scale features. This process continues until all the nodes in some match-set are matched (success) or until no more coarse level estimates remain (failure). In the remainder of this section we present the details of this matching procedure.

4.2.1. Coarse Scale Hypothesis Formation

The first step in the matching process is to match the model graph's root node with the coarsest level (16×16) of the input image's zero-crossing pyramid. The generalized Hough

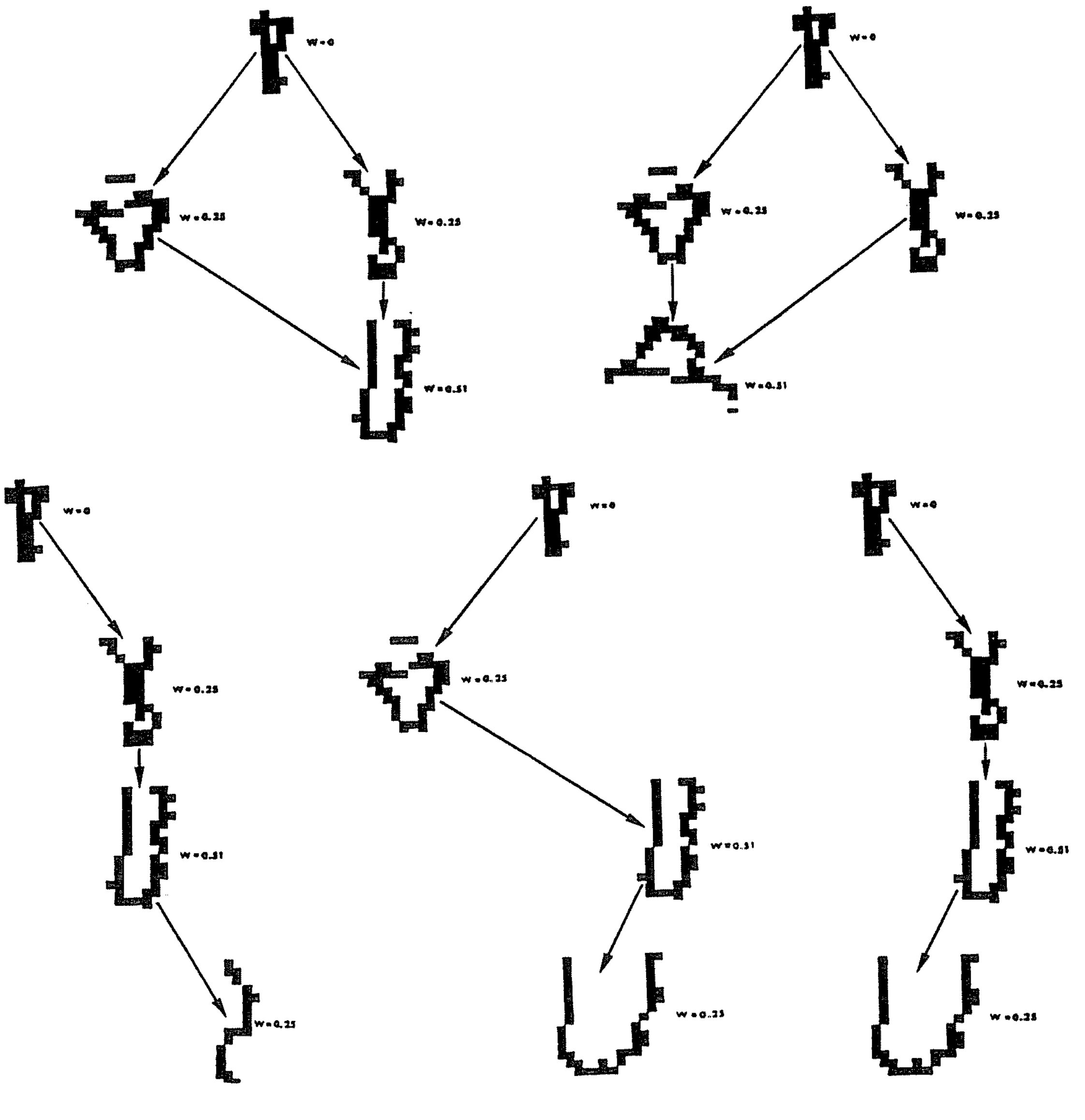

Figure 8. Several of the match-sets for the room key model graph indicating sufficient sub-graphs for successful matching.

transform is used for this purpose, resulting in a three-dimensional accumulator array of size $16 \times 16 \times 16$, representing two degrees of freedom in position and one degree of freedom in orientation. We estimate orientation at this level in increments of $360/16 = 22.5°$.

The peaks in the Hough space accumulator array indicate the best estimates of the position and orientation of the object. Non-maximum suppression is performed on the accumulator array using a $3 \times 3 \times 3$ neighborhood to aid in peak detection. Next, all peaks of size greater than a predefined threshold are sorted in descending order. Each peak, beginning with the highest, is now hypothesized to be the actual position and orientation of the object in the image.

4.2.2. Fine Scale Verification

Given a peak from the coarse matching process, the fine scale matching process calculates the position and orientation of the children of the root node. For each child of the root, a 16×16 window centered on its estimated position is extracted from its level in the input image's zero-crossing pyramid. Another generalized Hough transform is now applied using this window and the R-table associated with the current model node. The resultant accumulator array in this case is $16 \times 16 \times 11$. That is, orientation is quantized to 11 values at multiples of $4.5°$ centered around its parent's estimated orientation angle (which was quantized in steps of size $22.5°$).

The generalized Hough transform takes time proportional to the number of edge points in the window times the size of the node's R-table times the number of orientations. Consequently, our use of small 16×16 windows and small R-tables saves considerable time over the brute-force method of trying all possible positions and orientations at the current level.

Following the completion of the Hough transform, if the maximum value in the accumulator array divided by the size of the node's R-table is greater than a predefined threshold, then the window matches the current node. Otherwise, the match fails. If the match succeeds, then the weight associated with the current node is added to the current match-set total. If the total is now greater than 1, then the object has been detected; otherwise, the same procedure is used recursively to try and match all children of the current node (using the updated position and orientation associated with the maximum peak in the current node's accumulator array). It should be noted that the selection of a threshold which eliminates only the weakest matches is adequate because "false positives" at one node are quickly disproved by finer scale descendant nodes.

In general, the coarse scale nodes help to rapidly disprove false hypotheses and to make fine adjustments in the hypothesized position and orientation of the object. There is not usually enough information in the coarsest nodes to verify the presence of the object in the image. Also, coarse nodes may match similar features of the wrong object. A coarse representation of the shaft of a key, for example, may not match well with a washer in the image, thus quickly disproving this hypothesis, but it may match well with the shaft of a bolt and so cannot be used to verify the presence of the key in the image. If an hypothesis is correct, the coarse nodes do reduce the error in the object's position and orientation estimates.

The fine scale nodes generally serve to verify an hypothesis. Fine nodes rarely give false matches; in order for a fine node to match at all, the feature must be small (relative to the size of the image) and at roughly the correct orientation. If the features in the fine nodes are chosen judiciously, this is unlikely to happen except with the correct hypothesis. As a result of these observations, the weights associated with nodes in a model graph should be chosen so that match-sets include mostly coarse and intermediate level nodes plus a few fine nodes to complete the verification of at least selected parts of the object.

In our prototype system, we have used a breadth-first strategy to search the model-graph, but any graph search algorithm can be used. If a node has been correctly matched, its children are placed at the end of a queue and the next node in the queue is tested. Child nodes inherit the centroid and orientation estimates of their parents, appropriately scaled to the sample rate of the child. When all the nodes in a match-set are correctly matched, their estimates of the centroid and orientation are compared to make sure they agree to within a predefined error tolerance. If they agree, the object is successfully detected.

If no match-set is successfully matched, the current hypothesis fails and the next highest peak from the root node's list becomes the new hypothesis. Hypotheses that have failed are never retried. This sequence of trying and discarding hypotheses continues until one hypothesis results in the matching of a match-set, or until there are no more hypotheses.

4.3. Results

Table 1 shows the results of the procedure on three test images containing keys, washers and screws. Image 1 is shown in Figure 6, Image 2 is shown in Figure 9 and Image 3 is shown in Figure 10. The model graph shown in Figure 7 was used to recognize the room key in each image (the upper-right key in Figure 6). The total size of all R-tables in the model is 295. With both Images 2 and 3, the first hypothesis was the correct one and the matching process examined five nodes, with one failure, to find the key. The failure node in both cases contained a feature which was occluded. In the more complicated image, Image 1, the system had to try ten different hypotheses before finding the correct one. The strongest hypothesis had a peak size of 0.84 while the correct hypothesis had a peak size of 0.77. For Image 1, twenty-eight nodes were tested before a match-set was completely matched.

To roughly estimate the speedup of this method over a single scale generalized Hough approach, consider Image 1. There were about 100 edge points in every window selected from Image 1's pyramid. Thus the total computation time for this example was proportional to $295 \times 11 \times 100 \approx 3 \times 10^5$. In contrast, using the generalized Hough transform with the finest scale image (128×128) and a fine scale R-table for the key, the time would be proportional to 1400 (the number of edge points in the image) $\times$ 312 (the size of the R-table) $\times$ 80 (= 360/4.5 possible orientations), or about 3×10^7. Thus the speedup is two orders of magnitude for this test image.

Image	No. hypotheses tested	No. nodes tested	Avg. No. nodes to disprove a false hypothesis
1	10	28	3.1
2	1	5	0
3	1	5	0

Table 1. Results of the algorithm for three test images.

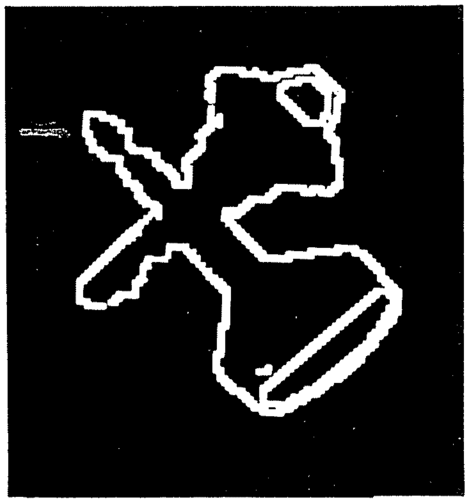

Figure 9. Test image 2.

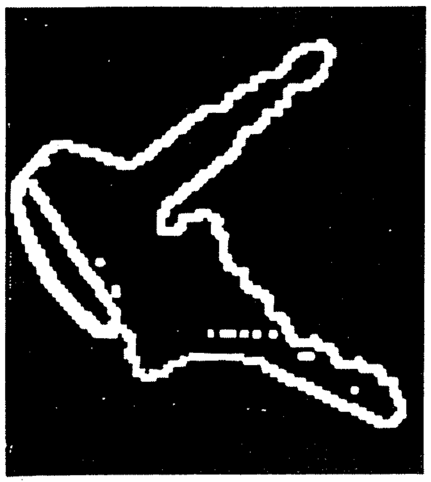

Figure 10. Test image 3.

5. Multiscale Texture Segmentation

In this section we describe a multiscale linking algorithm for texture segmentation. It is based on the computation of spatial properties of long, straight edge segments at fixed orientations. Features are computed for each edge segment in terms of the distances to the nearest neighboring edge segments of given orientations. This produces a set of sparse edge separation maps of features which are then used as the basis of a pyramid linking procedure for hierarchically grouping edges into homogeneously textured regions. Segmentation is performed in one bottom-up pass of linking nodes to their most similar parent. Thus each level of the pyramid corresponds to a particular scale of blurring the texture feature values, as determined by the linking process.

An important contribution of this section is to show how multiscale techniques can be effectively used with sparse iconic data. The method presented is based on the detection and grouping of long, straight edge segments in order to segment an image into regions containing different textures. Because edge segments are relatively sparse, yet a final partition of the image is required to define a complete segmentation, we needed to develop new techniques for both grouping sparse edge properties and interpreting the areas between edge tokens. Our previous work showed that using spatial properties of edge segments as features for texture classification is very successful (Kjell and Dyer, 1985). This section shows that similar features can also be used successfully for texture segmentation. Section 5.1 describes the steps associated with the detection of the extended edge segments and the computation of inter-edge separation features in the original (fine scale) image. Section 5.2 presents the pyramid linking algorithm which groups similar features at coarser and coarser scales using the upper levels of the pyramid. Section 5.3 summarizes the results of the algorithm on several different textures.

5.1. Edge Separation Features

The basis of our method is to use extended edge segments only as the tokens for texture segmentation. In this section we describe briefly a procedure for detecting extended edges and then computing several inter-segment spatial separation features. More details on this procedure are given in (Kjell and Dyer, 1986, 1987).

The goal of the edge detection step is to find long, straight edges of well-defined orientations. These extended edges are tokens more complex than pixels since they are built up in several steps from evidence collected over large areas of the image. Since the texture method will measure spatial relations between these edge-based tokens, for simplicity we will detect extended edge segments, each of which has a single global orientation. Hence in contrast to many edge detectors which attempt to follow contours around corners, our detector will find segments which are approximately straight and oriented in one of eight fixed orientations, $45i^\circ$, for $i = 0, \ldots, 7$.

Briefly, extended edges are detected as follows. Eight directional Laplacian-of-Gaussian operators are applied to the original image and zero-crossings are then detected in each result. Each zero-crossing map is then processed independently, removing weak edge points and linking adjacent edge points together to form approximately straight extended edge segments in the orientation associated with map's operator. This process fills small gaps between edge points and also removes short segments. The final output of this stage is eight maps of extended edges, a map for each orientation. Each map contains edges of a single orientation and each extended edge is a connected chain of pixels. Notice that a single pixel may be part of several extended edges, each in a different orientation map. These eight maps are the only information used by the texture method described below. Hence grayscale intensity values are not used; we only require extended edge maps representing sets of edge segments which are thin and approximately straight.

Figure 11 shows an image containing two textures, raffia and pigskin. Figure 12 is an image containing straw and water texture regions. The 0° extended edge segments detected in the Raffia/Pigskin image are shown in Figure 13.

Following the detection of extended edge segments, we next compute a set of spatial properties of these edges. For simplicity we have investigated statistics of *closest pairs of extended edge segments*. That is, for each extended edge segment we compute the value d_i which is the average distance from the given segment to the nearest segment of orientation $45i$°. Hence associated with each edge segment is a vector of eight spatial feature values. A total of 64 different texture features are computed. See (Kjell and Dyer, 1986, 1987) for more details on this procedure.

5.2. Segmentation

Most previous methods for texture segmentation have used pixel grayscale properties or features based on the gray levels in a small block centered at each pixel. Our method is based on higher level symbolic tokens and hence does not compute a feature for each pixel. Our goal is to group extended edges into distinct textural regions by merging areas of the image into regions containing similar extended edge features. Each region will be represented by an average edge separation vector over all of the extended edges contained in the region. There are two major difficulties with this type of approach. First, regions must be carefully grown so that extended edges from more than one texture are not included in the same region. Second, the method must be computationally feasible — that is, exponential time solutions which consider all possible combinations of edge segment groupings are not practical.

In this section we define a class of methods that (a) segments images to pixel level precision using extended edge segment features instead of pixel properties, (b) uses spatial context of regions of uniform texture, (c) does not use training data, (d) computes texture properties at multiple scales and (e) iteratively merges regions in parallel so that regions can grow in size at an exponential rate. The algorithm is based on Burt's overlapped pyramid structure to impose a fixed local control strategy for hierarchical region growing (Burt *et al.*, 1981). In addition, we use the bottom-up, iterative linking method in (Cibulskis and Dyer, 1984a, 1984b).

As discussed earlier, this approach simultaneously computes coarser and coarser scale descriptions while locally merging similar regions into larger and larger regions. The algorithm iterates at a given level until all nodes are linked to their best parent node at the next level. This process is then repeated at the next higher level, and so on until one pass up the pyramid is completed. In the algorithm used here, nodes are always forced to choose a best parent. This insures that there will be at most four regions represented at the level below the apex. All input images are assumed to contain exactly two distinct textures, so we then group these four or fewer regions into the best two regions.

The finest scale description to this procedure is a sparse set of texture separation features. That is, one or more edge separation maps are used to define a vector of texture feature values at each edge pixel. Note that most pixels will have an undefined value since only pixels which are part of an extended edge will have a defined value in an edge separation map. If several edge separation maps are used, the vector for a pixel may have one, several, or all values undefined.

In initial experiments it was found that when most pixels have undefined values, the early stages of linking were very unreliable. Consequently, rather than use the raw edge separation values, we first smoothed these values using a Gaussian-like kernel (using a small generating kernel and three or four of the bottom levels of the pyramid to efficiently perform this operation). This step serves two purposes: (a) most pixels at the bottom level which are not part of an extended edge will have their undefined values replaced by values representing the smoothed averages of their nearest extended edges and (b) pixels at the bottom level which are part of an extended edge will have their separation values smoothed using the separation values of other

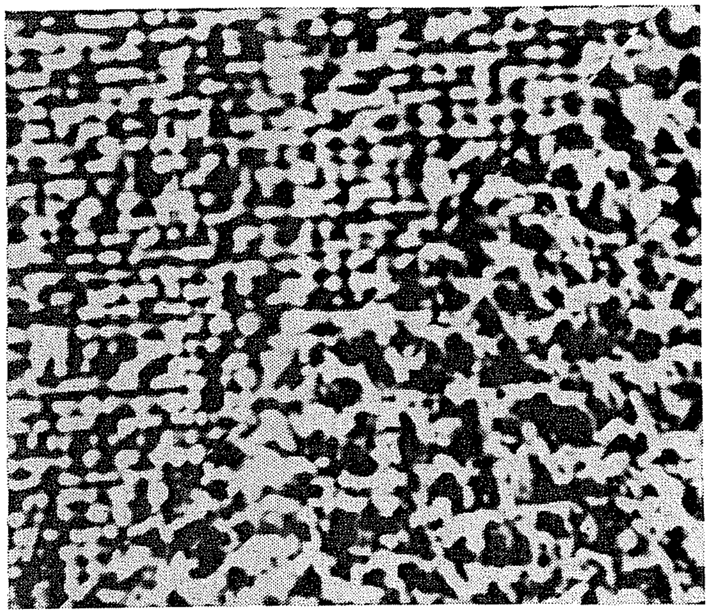

Figure 11. Raffia/Pigskin texture image.

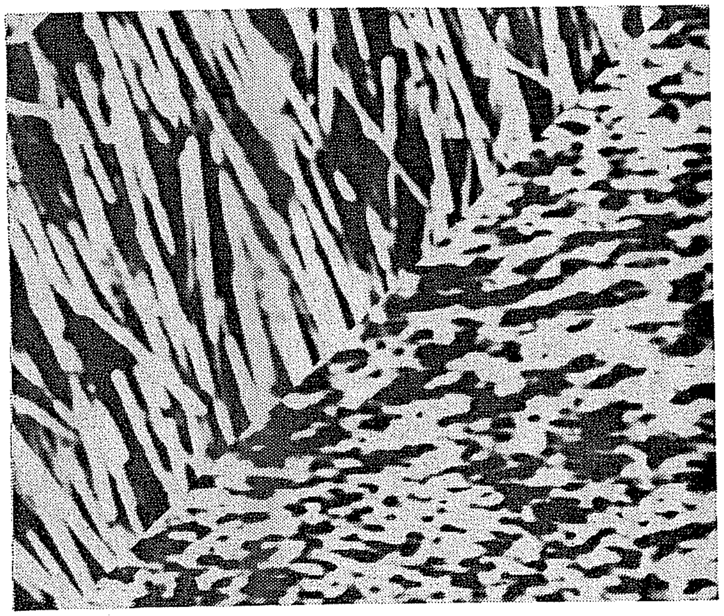

Figure 12. Straw/Water texture image.

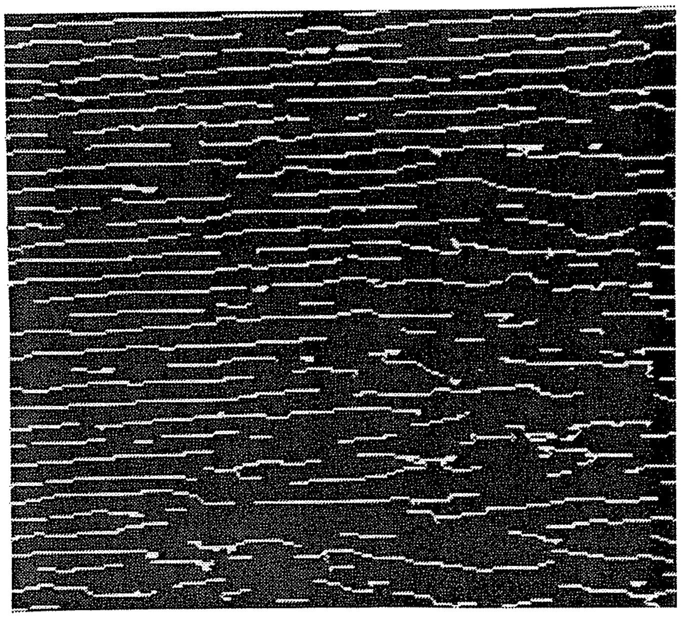

Figure 13. Extended edges in direction 0° for Raffia/Pigskin image.

nearby edge points. Figure 14 shows the result of this smoothing process for the Raffia/Pigskin image. 0° edge points were initially labeled with their average distance to their nearest 180° edges and then smoothed using the equivalent of a 46×46 kernel.

The smoothed edge separation feature vectors define the initial values for the finest scale texture description of an input image. At each level above the base, a coarser description is computed by averaging the feature vectors of all children pixels currently linked to the given pixel. Undefined values are not used in computing the average; if no children have a defined value for a particular feature, that feature is assigned an undefined value.

The linking rule was modified somewhat from that given in (Cibulskis and Dyer, 1984b) because many feature values are still undefined in our case. If a single feature value is used, the child chooses the parent pixel closest in value to its own. If the child's value is undefined, it picks arbitrarily any parent with a defined value. If no parent has a defined value, the choice is also arbitrary. If several feature values are used, the closest parent is the one with the least average difference between defined property values.

5.3. Results

Figure 15 shows the segmentation of the Straw/Water image using the algorithm with the single texture feature specifying the smoothed separation values between 0° and 180° edge segments. 97% of the pixels were classified correctly in this image. Table 2 summarizes the segmentation accuracies for both the Straw/Water and Raffia/Pigskin images using sixteen individual features, three sets of vectors containing four features and one vector containing all sixteen features.

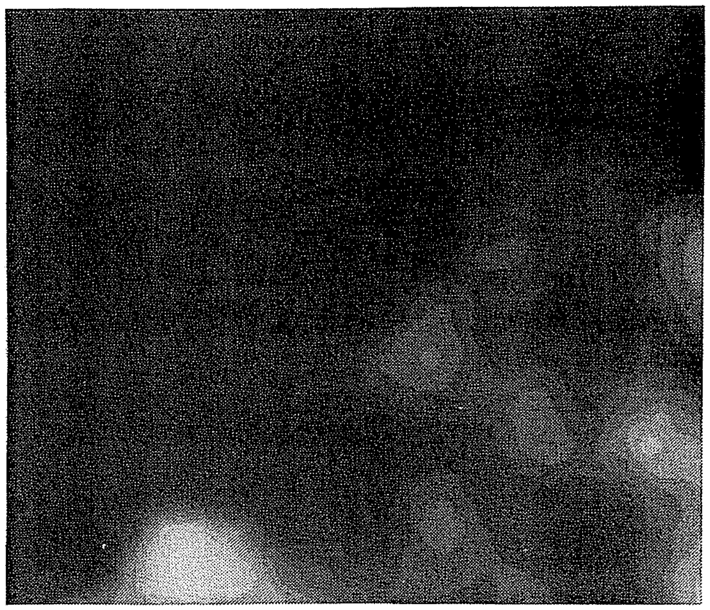

Figure 14. Smoothed separation map for the Raffia/Pigskin image using the 0-180 edge separation map.

As a final example, Figure 16 is a close-up image of a raccoon lying in dried grass; the upper-left part of the image is the raccoon's fur and the lower-right part is the grass. The fur of the raccoon and the blades of the grass are both linear textures, but with different orientations. The orientation of the fur is mostly left-leaning vertical while the grass orientation is primarily horizontal. Based on this observation, two oriented edge separation features were chosen: the 0°-180° separation between antiparallel horizontal edges and the 135°-315° separation between antiparallel, left-leaning vertical edges. The resulting segmentation is shown in Figure 17.

Edge Orientations	Straw/Water Image	Raffia/Pigskin Image
0-180	97.1	91.8
0-270	79.1	65.6
45-225	91.9	76.8
45-315	68.6	76.1
90-0	91.6	96.0
90-270	86.0	68.2
135-45	83.7	73.7
135-315	80.4	65.0
180-0	86.6	91.2
180-90	70.2	70.4
225-45	91.6	74.3
225-135	59.0	77.0
270-90	82.8	68.0
270-180	94.2	93.6
315-135	77.7	88.3
315-225	83.2	61.0
4 anti-parallel	95.9	85.2
4 best	96.5	91.5
4 worst	75.1	75.3
all 16	95.6	93.9

Table 2. Percentages of pixels segmented correctly for Straw/Water image and Raffia/Pigskin image. First column indicates which smoothed inter-edge separation features were used. The first sixteen are single texture features; the next three are selected quadruples of texture features; the last case used a feature vector of length 16 containing all the separation values for the sixteen pairs shown at the top of the table.

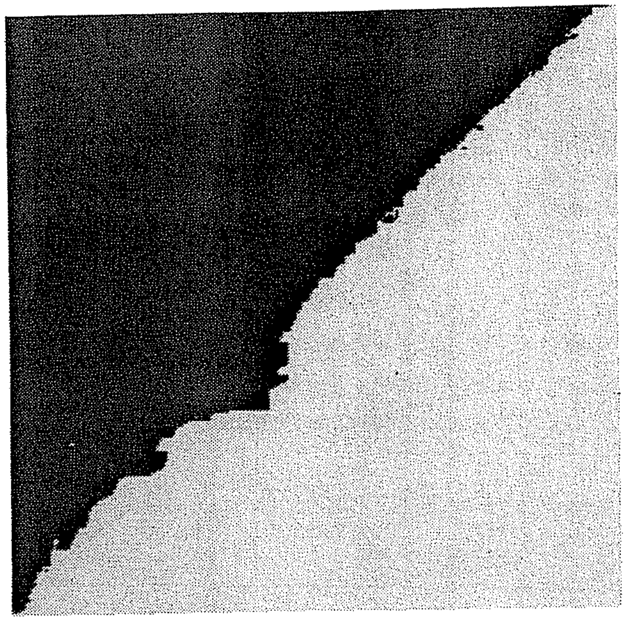

Figure 15. Segmentation of the Straw/Water image using the smoothed 0-180 edge separation map.

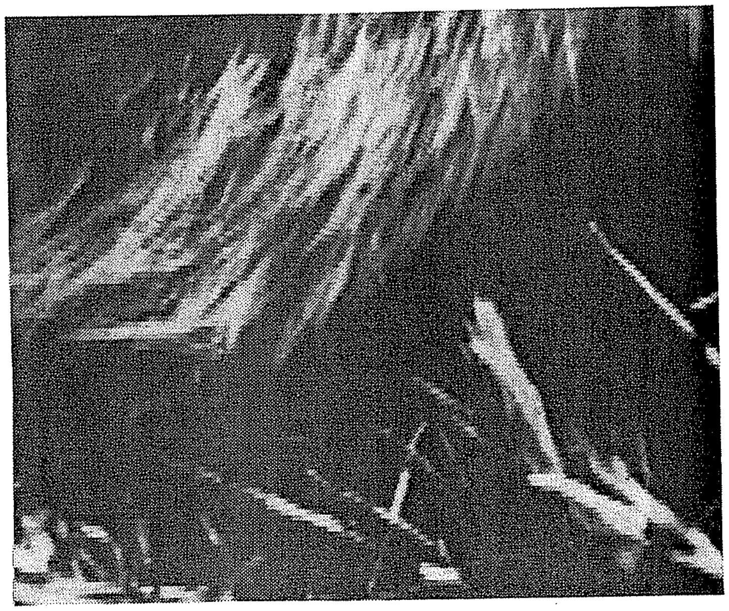

Figure 16. Raccoon lying in grass.

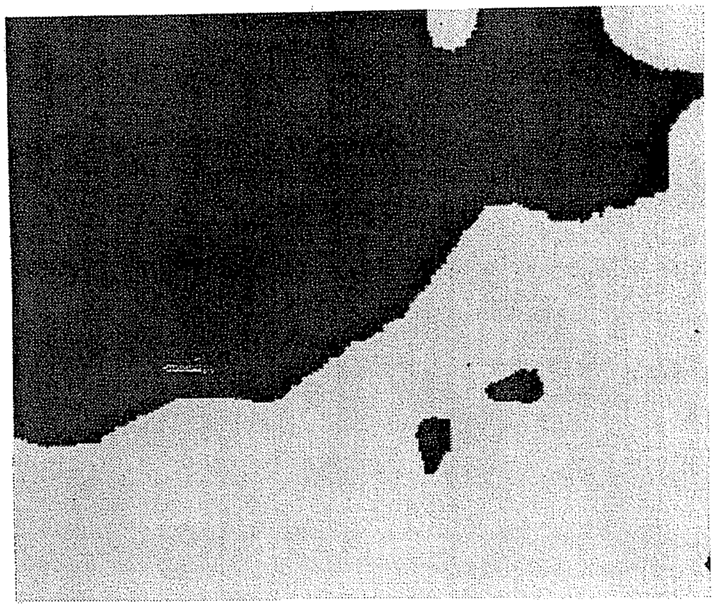

Figure 17. Segmentation of the Raccoon/Grass image using two edge separation features.

References

Anderson, C. H., Burt, P. J., and VanderWal, G. S., Change detection and tracking using pyramid transform techniques, *Proceedings of the Society of Photo-Optical Instrumentation Engineers Conference on Intelligent Robots and Computer Vision* **579**, 1985, 72-78.

Antonisse, H. J., Image segmentation in pyramids, *Comput. Gr. Image Process.* **19**, 1982, 367-383.

Asada, H., and Brady, M., The curvature primal sketch, *IEEE Trans. Pattern Anal. Mach. Intell.* **8**, 1986, 2-14.

Babaud, J., Witkin, A. P., Baudin, M., and Duda, R. O., Uniqueness of the Gaussian kernel for scale-space filtering, *IEEE Trans. Pattern Anal. Mach. Intell.* **8**, 1986, 36-33.

Ballard, D. H., Generalizing the Hough transform to detect arbitrary shapes, *Pattern Recognition* **13**, 1981, 111-122.

Ballard, D. H., Parameter nets, *Artif. Intell.* **22**, 1984, 235-267.

Baugher, E. S., and Rosenfeld, A., Boundary localization in an image pyramid, Technical Report CS-TR-1488, Center for Automation Research, University of Maryland, May 1985.

Bolles, R. C., and Cain, R. A., Recognizing and locating partially visible objects: The local-feature-focus method, *Int. J. Robotics Res.* **1**, 1982, 57-82.

Borgefors, G., On hierarchical edge matching in digital images using distance transformations, Ph.D. dissertation, Royal Institute of Technology, Stockholm, Sweden, 1986.

Burt, P. J., Fast filter transforms for image processing, *Comput. Gr. Image Process.* **16**, 1981, 20-51.

Burt, P. J., Hong, T. H., and Rosenfeld, A., Segmentation and estimation of image region properties through cooperative hierarchical computation, *IEEE Trans. Syst. Man Cybern.* **11**, 1981, 802-809.

Burt, P. J., Fast algorithms for estimating local image properties, *Comput. Vision, Gr. Image Process.* **21**, 1983, 368-382.

Burt, P. J., and Adelson, E. H., The Laplacian pyramid as a compact image code, *IEEE Trans. Comm.* **31**, 1983, 532-540.

Burt, P. J., The pyramid as a structure for efficient computation, in *Multiresolution Image Processing and Analysis*, A. Rosenfeld (Ed.), Berlin: Springer-Verlag, 1984, 6-35.

Cibulskis, J. M., and Dyer, C. R., Node linking strategies in pyramids for image segmentation, in *Multiresolution Image Processing and Analysis*, A. Rosenfeld (Ed.), Berlin: Springer-Verlag, 1984, 109-120. (a)

Cibulskis, J. M., and Dyer, C. R., An analysis of node linking in overlapped pyramids, *IEEE Trans. Syst. Man Cybern.* **14**, 1984, 424-436. (b)

Clark, J. J., and Lawrence, P. D., A theoretical basis for diffrequency stereo, *Comput. Vision, Gr. Image Process.* **35**, 1986, 1-19.

Crowley, J. L., and Parker, A. C., A representation for shape based on peaks and ridges in the difference of low-pass transform, *IEEE Trans. Pattern Anal. Mach. Intell.* **6**, 1984, 156-169.

Crowley, J. L., and Stern, R. M., Fast computation of the difference of low-pass transform, *IEEE Trans. Pattern Anal. Mach. Intell.* **6**, 1984, 212-222.

Crowley J. L., and Sanderson, A. C., Multiple resolution representation and probabilistic matching of 2-D gray scale shape, *Proceedings of the Workshop on Computer Vision: Representation and Control*, 1984, 95-105.

Crowley, J. L., A multiresolution representation for shape, in *Multiresolution Image Processing and Analysis*, A. Rosenfeld (Ed.), Berlin: Springer-Verlag, 1984, 169-189.

Dyer, C. R., Rosenfeld, A., and Samet, H., Region representation: boundary codes from quadtrees, *Comm. ACM* **23**, 1980, 171-179.

Dyer, C. R., A VLSI pyramid machine for hierarchical parallel image processing, *Proceedings of the Pattern Recognition and Image Processing Conference*, 1981, 381-386.

Fleck, M. M., Local rotational symmetries, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1986, 332-337.

Glazer, F., Reynolds, G., and Anandan, P., Scene matching by hierarchical correlation, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1983, 432-441.

Glazer, F., Multilevel relaxation in low-level computer vision, in *Multiresolution Image Processing and Analysis*, A. Rosenfeld (Ed.), Berlin: Springer-Verlag, 1984, 312-330.

Grimson, W. E. L., A computer implementation of a theory of human stereo vision, *Phil. Trans. Royal Soc. London B* **292**, 1981, 217-253.

Grimson, W. E. L., *From Images to Surfaces: A Computational Study of the Human Early Visual System* Cambridge, Mass.: MIT Press, 1981.

Hanson, A. R., and Riseman, E. M., Segmentation of natural scenes, in *Computer Vision Systems*, A. Hanson and E. Riseman (Eds.), New York: Academic Press, 1978, 129-174.

Hanson, A. R., and Riseman, E. M., Processing cones: a computational structure for image analysis, in *Structured Computer Vision*, S. Tanimoto and A. Klinger (Eds.), New York: Academic Press, 1980, 101-131.

Hartley, R., A Gaussian-weighted multiresolution edge detector, *Comput. Vision, Gr. Image Process.* **30**, 1985, 70-83. (a)

Hartley, R., Segmentation of optical flow fields by pyramid linking, *Pattern Recogn. Letters* **3**, 1985, 253-262. (b)

Hartley, R., and Rosenfeld, A., Hierarchical line linking for corner detection, in *Integrated*

Technology for Parallel Image Processing, S. Levialdi (Ed.), New York: Academic Press, 1985, 101-119.

Hong, T. H., Narayanan, K., Peleg, S., Rosenfeld, A., and Silberberg, T., Image smoothing and segmentation by multiresolution pixel linking: Further experiments and extensions, *IEEE Trans. Syst. Man Cybern.* **12**, 1982, 611-622. (a)

Hong, T. H., Shneier, M., and Rosenfeld, A., Border extraction using linked edge pyramids, *IEEE Trans. Syst. Man Cybern.* **12**, 1982, 660-668. (b)

Hong, T. H., and Shneier, M., Extracting compact objects using linked pyramids, *Proceedings of the Image Understanding Workshop*, 1982, 58-71.

Hong, T. H., and Rosenfeld, A., Unforced image partitioning by weighted pyramid linking, *Proceedings of the Image Understanding Workshop*, 1982, 72-78.

Hong, T. H., Shneier, M., Hartley, R., and Rosenfeld, A., Using pyramids to detect good continuation, *IEEE Trans. Syst. Man Cybern.* **13**, 1983, 631-635.

Hong, T. H., and Rosenfeld, A., Compact region extraction using weighted pixel linking in a pyramid, *IEEE Trans. Pattern Anal. Mach. Intell.* **6**, 1984, 222-229.

Hong, T. H., and Shneier, M., Extracting compact objects using linked pyramids, *IEEE Trans. Pattern Anal. Mach. Intell.* **6**, 1984, 229-237.

Kelly, M. D., Edge detection in pictures by computer using planning, in *Machine Intelligence*, 6, B. Meltzer and D. Michie (Eds.), Edinburgh: Edinburgh University Press, 1971, 397-409.

Kjell, B. P., and Dyer, C. R., Edge separation and orientation texture measures, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1985, 306-311.

Kjell, B. P., and Dyer, C. R., Segmentation of textured images, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1986, 476-481.

Kjell, B. P., and Dyer, C. R., Segmentation of textured images using pyramid linking, to appear in *Pyramidal Systems for Image Processing and Computer Vision*, V. Cantoni and S. Levialdi (Eds.), Berlin: Springer-Verlag, 1987.

Kropatsch, W. G., Hierarchical curve representation in a new pyramid scheme, Technical Report CS-TR-1522, Center for Automation Research, University of Maryland, June 1985.

Levine, M. D., A knowledge-based computer vision system, in *Computer Vision Systems*, A. R. Hanson and E. M. Riseman (Eds.), New York: Academic Press, 1978, 335-352.

Levine, M. D., Region analysis using a pyramid data structure, in *Structured Computer Vision*, S. Tanimoto and A. Klinger (Eds.), New York: Academic Press, 1980, 57-100.

Liu, F-Y., Eastman, R., and Davis, L. S., Experiments in stereo matching using multiresolution local support, Technical Report CS-TR-1617, Center for Automation Research, University of Maryland, January 1986.

Marr, D., and Poggio, T., A computational theory of human stereo vision, *Proc. Royal Society of*

London B **204**, 1979, 301-328.

Marr, D., and Hildreth, E., Theory of edge detection, *Proc. Royal Society of London B* **207**, 1980, 187-217.

Marr, D., *Vision*, San Francisco: Freeman, 1982.

Miller, R., and Stout, Q. F., Pyramid computer algorithms for determining geometric properties of images, *Proceedings of the Symposium on Computational Geometry*, 1985, 263-271.

Mokhtarian, F., and Mackworth, A., Scale-based description and recognition of planar curves and two-dimensional shapes, *IEEE Trans. Pattern Anal. Mach. Intell.* **8**, 1986, 34-43.

Moravec, H. P., Towards automatic visual obstacle avoidance, *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, 1977, 584.

Moravec, H. P., Obstacle avoidance and navigation in the real world by a seeing robot rover, Technical Report CMU-RI-TR-3, Robotics Institute, Carnegie-Mellon University, Pittsburgh, PA, 1980.

Mori, K., Kidodi, M., and Asada, H., An iterative prediction and correction method for automatic stereo comparison, *Comput. Gr. Image Process.* **2**, 1973, 393-401.

Narayanan, K. A., Approximation of waveforms and contours by one-dimensional pyramid linking, *Pattern Recogn.* **15**, 1982, 389-396.

Narayanan, K. A., O'Leary, D. P., and Rosenfeld, A., Multi-resolution relaxation, *Pattern Recogn.* **16**, 1983, 223-230.

Neveu, C. F., Dyer, C. R., and Chin, R. T., Two-dimensional object recognition using multiresolution models, *Comput. Vision, Gr. Image Process.* **34**, 1986, 52-65.

Peleg, S., Naor, J., Hartley, R., and Avnir, D., Multiple resolution texture analysis and classification, *IEEE Trans. Pattern Anal. Mach. Intell.* **6**, 1984, 518-523.

Pietkainen, M., and Rosenfeld, A., Image segmentation by texture using pyramid node linking, *IEEE Trans. Syst. Man Cybern.* **11**, 1981, 822-825.

Pietkainen, M., and Rosenfeld, A., Gray level pyramid linking as an aid in texture analysis, *IEEE Trans. Syst. Man Cybern.* **12**, 1982, 422-429.

Rosenfeld, A., and Thurston, M., Edge and curve detection for visual scene analysis, *IEEE Trans. Comput.* **20**, 1971, 562-569.

Rosenfeld, A., and VanderBrug, G. J., Coarse-fine template matching, *IEEE Trans. Syst. Man Cybern.* **7**, 1977, 104-107.

Rosenfeld, A. (Ed.), *Multiresolution Image Processing and Analysis*, Berlin: Springer-Verlag, 1984.

Rosenfeld, A., Some pyramid techniques for image segmentation, Technical Report CS-TR-1664, Center for Automation Research, University of Maryland, May 1986.

Rosenfeld, A., Pyramid algorithms for perceptual organization, to appear in *Pyramidal Systems for Image Processing and Computer Vision,* V. Cantoni and S. Levialdi (Eds.), Berlin: Springer-Verlag, 1987.

Sabbah, D., Computing with connections in visual recognition of Origami objects, *Cognitive Science* **9**, 1985, 25-50.

Samet, H., The quadtree and related hierarchical data structures, *ACM Comput. Surv.* **16**, 1984, 187-260.

Shneier, M., Two hierarchical linear feature representations: edge pyramids and edge quadtrees, *Comput. Gr. Image Process.* **17**, 1981, 211-224.

Silberberg, T., Peleg, S., and Rosenfeld, A., Multi-resolution pixel linking for image smoothing and segmentation, *Proceedings of the Image Understanding Workshop*, 1981, 32-38.

Stout, Q. F., Sorting, merging, selecting, and filtering on tree and pyramid machines, *Proceedings of the International Conference on Parallel Processing*, 1983, 214-221.

Tanimoto, S. L., and Pavlidis, T., A hierarchical data structure for picture processing, *Comput. Gr. Image Process.* **4**, 1975, 104-119.

Tanimoto, S. L., Pictorial feature distortion in a pyramid, *Comput. Gr. Image Process.* **5**, 1976, 333-352.

Tanimoto, S. L., Template matching in pyramids, *Comput. Gr. Image Process.* **16**, 1981, 356-369.

Tanimoto, S. L., A pyramidal approach to parallel processing, *Proceedings of the 10th Annual International Symposium on Computer Architecture*, 1983, 372-378.

Terzopoulos, D., Multilevel computational processes for visual surface reconstruction, *Comput. Vision, Gr. Image Process.* **24**, 1983, 52-96.

Terzopoulos, D., Multilevel reconstruction of visual surfaces: variational principles and finite-element representations, in *Multiresolution Image Processing and Analysis*, A. Rosenfeld (Ed.), Berlin: Springer-Verlag, 1984, 237-310.

Terzopoulos, D., Image analysis using multigrid relaxation methods, *IEEE Trans. Pattern Anal. Mach. Intell.* **8**, 1986, 129-139.

Torre, V., and Poggio, T. A., On edge detection, *IEEE Trans. Pattern Anal. Mach. Intell.* **8**, 1986, 147-163.

Uhr, L., Layered "recognition cone" networks that preprocess, classify, and describe, *IEEE Trans. Comput.* **21**, 1972, 758-768.

Wells, W. M., Efficient synthesis of Gaussian filters by cascaded uniform filters, *IEEE Trans. Pattern Anal. Mach. Intell.* **8**, 1986, 234-239.

Werman, M., and Peleg, S., Min-max operators in texture analysis, *IEEE Trans. Pattern Anal. Mach. Intell.* **7**, 1985, 730-733.

Williams, L. R., and Anandan, P., A coarse-to-fine control strategy for stereo and motion on a mesh-connected computer, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1986, 219-226.

Wilson, H. R., and Giese, S. C., Threshold visibility of frequency gradient patterns, *Vision Res.* **17**, 1977, 1177-1190.

Wilson, H. R., and Bergen, J. R., A four mechanism model for spatial vision, *Vision Res.* **19**, 1979, 19-32.

Witkin, A. P., Scale-space filtering, *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, 1983, 1019-1021.

Witkin, A. P., Terzopoulos, D., and Kass, M., Signal matching through scale space, *Proceedings of the 5th National Conference on Artificial Intelligence*, 1986, 714-719.

Wong, R. Y., and Hall, E. L., Sequential hierarchical scene matching, *IEEE Trans. Comput.* **27**, 1978, 359-366.

Yuille, A. L., and Poggio, T. A., Fingerprint theorems, *Proceedings of the National Conference on Artificial Intelligence*, 1984, 362-365.

Yuille, A. L., and Poggio, T. A., Scaling theorems for zero crossings, *IEEE Trans. Pattern Anal. Mach. Intell.* **8**, 1986, 15-25.