Contour Motion Compensation for Image Sequence Coding

by

Chong Uk Lee

B.S.E.E. Massachusetts Institute of Technology (1981)
M.S.E.E. Massachusetts Institute of Technology (1985)

Submitted in Partial Fulfillment
of the Requirement for the
Degree of

Doctor of Philosophy

at the

Massachusetts Institute of Technology

May 1989

©1989, Massachusetts Institute of Technology

Signature of the Author _____
Department of Electrical Engineering and Computer Science
May 22, 1989

Certified by _____
William F. Schreiber
Thesis Supervisor

Accepted by _____
Arthur C. Smith
Chairman, EE Department Committee on Graduate Students

MASS. INST. TECH.
JUL 11 1989
LIBRARIES

ARCHIVES

Contour Motion Compensation for Image Sequence Coding

by

Chong Uk Lee

Submitted to the Department of Electrical Engineering and Computer Science on May 22, 1989 in partial fulfillment of the requirements for the Degree of Doctor of Philisophy in Electrical Engineering

Abstract

In this thesis we develop contour matching and interpolation techniques for the purpose of motion compensation of image sequences. Contours extracted from an image segmentation process over a sequence of frames are the input data. The moving contours are matched from frame to frame and are interpolated to produce intermediate frames. Two matching techniques are developed; The first is a transform technique based on the circular transform of the contour, similar to the Fourier transform. Applicable only to closed contours, the transform and the associated normalization procedure extract the shape parameters of the contour. A mean square error criterion is used to match pair of contours. An associated interpolation technique is also developed that interpolates the intermediate contour in the shape space and inverse transforms it for reconstruction. The second technique is a spatial method applicable to open contours. By extracting the curvature or the tangent of the contour, a distance matrix is set up. Imposing a rigid motion constraint enables a simple formulation of a Hough transform technique that estimates the alignment parameters. The procedure finds the transformation parameters as well as the end points of the matching segments. However, this matching technique lacks a suitable interpolation and reconstruction method. Both the transform and the spatial techniques are shown to be relatively robust when given a contour sequence extracted from a set of simple gray scale images. The motion compensation of a moving contour sequence is demonstrated using the two techniques.

Thesis Supervisor: William F. Schreiber
Title: Professor of Electrical Engineering

Acknowledgments

I am very thankful to Prof. William F. Schreiber whose continuous support, supervision, and encouragements made this thesis possible. I am also grateful to the members of the ATRP group who have created a pleasant research environment. A special thank goes to John Wang for his unending supply of computer expertise and wisdom. I wish to recognize all my friends at MIT who made my stay at MIT very memorable. There are many people, relatives and acquaintances, who provided valuable moral support. I am very grateful to my Mother and express my greatest love and respect. She was behind me through out my education, always with love and care.

This work was supported in part by the sponsors of the Advanced Television Research Project.

Contents

List of Figures

CHAPTER 1. INTRODUCTION

Most digital image processing techniques are based on the canonic representation of the image as a two-dimensional array of intensity samples. The intensity of a sample is usually quantized so that the image can be stored and transmitted by a finite number of bits. The goal of a conventional coding scheme is then to reduce the number of bits to represent the image by exploiting the property of local correlation among neighboring samples. Therefore the traditional view treated an image as a two-dimensional stochastic signal and information theory played a central role in increasing the coding efficiency within this framework. However in the recent years, it became apparent that the coding efficiency, as measured in terms of compression ratio, has been approaching the limit. A higher compression ratio could be obtained only at the expense of a rapid degradation of the image quality, where the type and severity of the degradation depends on the coding scheme used. To further increase the compression ratio it seems necessary to exploit the properties of the human visual system to a fuller extent since the eye is the ultimate receiver of the processed image.

It is clear that the low-level processing that goes on in the human visual system has no direct equivalence of the two-dimensional sampling grid the current image processing is based on. Rather, the eye is more adapted to deal with rather abstract features such as edges and lines through the lateral inhibition process that occurs at the retina and the orientation sensitive cells in the visual cortex [7]. For example, the eye can make out and understand most of the objects even from a high-pass filtered image because the high-frequency information usually corresponds to edges in the image. On the other hand, the intermediate level processing that occurs in the brain seems to involve topological characterization such as shapes, blobs, regions, etc., which can be derived from the edge information supplied by the low-level processing. It would come as a no surprise that so much can be conveyed by simple line drawings such as caricatures and cartoons. These low and intermediate processings can be viewed as data

reduction steps to a higher-level processing which puts together the shape information to recognize three-dimensional objects in the image.

Although the above view of the human visual system is very simplistic and is undoubtedly based on inadequate understanding of the actual process, it provides us with a new framework in processing images for the purpose of human viewing. In the framework of region-based image model, the image is represented by a collection of regions where the region boundary is represented by the bounding contour and inside the region is described by its texture. This approach corresponds to the intermediate level of processing described above. At this level of abstraction the image can be represented as concisely as possible without having to introduce knowledge as to how the image is formed (from the three-dimensional real world) or knowledge as to how the human visual system perceives objects.

The term region is not very easy to define quantitatively. That is one of the reasons that image segmentation is not a solved problem and still is an active area of research. In qualitative terms, a region can be defined as a spatially connected area of similar property such that crossing the region boundary leads into another region of different property. The property should be such that the boundary between regions is perceptible by the human eye. The word *texture* is generally used to refer to this property inside the region and we shall use the word *contour* to describe the boundary between regions.

The significance of representing images by contours and textures is the high degree of data reduction possible. The contour can be represented by a polygonal approximation which can be efficiently coded using techniques such as curve fitting, chain code, etc. A wealth of contour coding techniques exist under the general heading of outline coding which were originally developed for typographical character coding [34,35]. On the other hand the texture can be characterized as a low-contrast random detail. Because of its random nature it has been widely suggested that the texture need not be reproduced on a point-by-point basis and therefore can be characterized by a small set of

statistical properties. A synthetic texture reproduced this way can look remarkably similar to the real texture to a human eye [61]. An alternative approach that more or less reproduces texture on a point-by-point basis is to use traditional coding techniques such as predictive or transform techniques [58]. Here, the local statistics do not vary much over the region so that the information-theoretic approach will be quite valid and we can expect a high coding efficiency.

The recent interest and progress in image sequence coding has seen many coding schemes based on motion information. They make use of the high degree of correlation between successive frames in the sequence by measuring the motion information of one sample point in the frame to the corresponding point in the next frame. A motion vector describes the motion of a sample point from frame to frame. A typical system transmits fewer frames than there are in the original sequence, computes the motion vector fields, and interpolates the missing frames based on the transmitted frames and the motion vector fields. It is apparent that a successful contour-texture based coding scheme could be naturally extended to code sequences by measuring the motion of a region instead that of a sample point. The distinct advantages of doing so are numerous. A more robust and accurate motion estimation should be possible since the region has a distinct shape and texture which aids in finding the corresponding region. The region boundary will often coincide with the discontinuities in the motion vector field. It can be speculated that only the contour information will be enough in most cases and the texture can be used to resolve ambiguous cases. Although the operation involved will be more complicated than that of pixel-based motion-estimation techniques, the number of operations will be fewer since the number of contours will be fewer than the pixels in the image. In addition, the amount of motion information will be very small because only the correspondence information between regions is needed (of course the matter becomes more complicated when the region is allowed to deform over time). In addition, the problem of occlusion (regions

covering one another) and uncovered background, which is difficult to handle in the conventional approach, becomes easier to analyze.

On the reconstruction side, where more frames are interpolated from the transmitted frames using motion information, the contour-texture approach has more advantages: interpolating the contour is relatively simple and the texture can be synthesized by interpolating the texture parameters. It seems safe to make the assumption that the texture in a given region does not vary rapidly in time. Therefore, with a further assumption that a particular region remains in the image over a period of time, predictive coding of the parameters that describe the texture in the region will give a higher compression ratio. (Intuitively, we can imagine a cylinder traced out by the closed contour of a moving region. The two ends of this cylinder will correspond to the appearance and disappearance of the region due to various reasons such as occlusion at the image boundary or by foreground objects.)

In the current image-processing literature, there does not seem to be much attempt at combining the contour-texture based image coding with the motion-compensated sequence coding. Certainly there is a great deal of research done on individual processing steps necessary to achieve this. The three main areas are segmentation, contour motion compensation, and texture analysis. Segmentation has been widely studied by the scene-analysis and pattern-recognition community. Motion compensation has been mainly developed in the traditional image-processing community. Texture analysis has been researched by both. However, in view of the amount of research effort in each of the areas, putting all three techniques together and building a successful image coding system is indeed a great task. In this thesis we choose to concentrate on solving the least researched area, processing and motion compensating the contours.

We assume that a suitable segmentation process exists that extracts contours from an image sequence. Given a sequence of contour frames displaced in time, the task is then to identify the matching contours and to perform motion compensation to reconstruct

the contours for the in-between frames. The basic approach we employ starts by removing the orientation information from the contour until only the shape information of the contour remains. The shape matching and interpolation takes place in this *shape space*. Then, appropriate orientation information is added back to reconstruct the contour at a desired orientation and time displacement.

In Chapter 2, the background framework relevant to the contour-texture based image coding will be presented. Although a contour can be represented by a polygonal approximation, we will apply the signal processing concepts and view it as a contour *signal*. Based on a complex signal representation of the contour, we develop two distinct approaches in matching and interpolating contours. In Chapter 3, a Circular Transform technique similar to the Fourier transform is developed, and a frequency-domain interpretation is applied to match and interpolate *closed* contours. Because transform methods cannot handle open (partial) contours very easily, in Chapter 4 we develop a spatial-domain technique based on the distance matrix to match partial contours. Chapter 5 concludes with experimental results obtained using both techniques.

CHAPTER 2. BACKGROUND

One of the earliest and one of the most recent coding schemes based on contours are presented first to establish the background and the motivation for developing a contour-texture based coding system. A brief description of the current states of contour-based motion estimation and texture analysis follows. The issues of segmentation are presented in more detail since it is a necessary step before contours can be manipulated. Discussions of contour coding and shape analysis set the stage for the next two chapters on matching and interpolation of contours.

2.1 Synthetic Highs System

This is perhaps the earliest predecessor of the class of feature-based coding schemes which was developed in the 60's [1,2]. Like two-channel coding system [3], it achieves data compression by exploiting the spatial masking effect of the eye. The eye is less sensitive to the noise in the vicinity of the large discontinuities in brightness such as edges in the image. Therefore the edge information can be quantized coarsely to reduce the bit rate. Edges normally correspond to the high-frequency component of the image.

In the synthetic highs system the image is first filtered by a low-pass filter to obtain the low-frequency image. The low-frequency image is subsampled and coded in a conventional manner. The edge points are found using Laplacian or Gradient operators. The fact that the edge points line up and form contours is exploited to code the edge information as contours. The gradient information is also coded along with the contour. At the receiver, the high-frequency image is *synthesized* from the gradient along the contour by applying a synthesis filter. It has been shown that a unique synthesis filter can be derived from the low-pass filter to recover the missing high-frequency image exactly from the gradient image [4]. The original image is then reconstructed by adding the low-frequency image and the synthesized high-frequency image. The coding system becomes information lossy when some of the contours with low gradient are not transmitted. This is necessary to reduce the number of contours transmitted in order to achieve a high compression ratio. The information discarded this way can be viewed as texture and a possible way to synthesize the texture has been suggested in [5].

2.2 Region-Growing Based Coding

This is one of the most recent developments in feature-based coding, advanced mainly by Kunt and Kocher [7,17]. First the image is segmented into regions by a region-growing algorithm. Then the contour defining the boundary of the region and the texture describing the variation within the region are coded separately. Unlike the synthetic highs system, however, no gradient information is sent along the contour. Also the coding of texture eliminates the need for sending the low-frequency image.

In order to segment the image into a small number of regions, three steps are taken. The first is an inverse gradient filter that removes the local granularity while preserving the edges as well as possible. This is a kind of adaptive filtering based on the local contrast and needs to be applied iteratively to remove the granularity sufficiently for the next step. Next, a relatively simple grey-level based region-growing technique is applied. It is basically an adaptive threshold method with a fixed threshold interval. Pixels that fall within the interval are admitted into the region. The interval is moved up and down the scale to admit the pels into the region with the constraint that the previously intercepted pels remain in the region. The third step tries to eliminate the artifacts produced by the region-growing algorithm, such as open contours and contours that are two pels wide, by heuristic rules.

After segmentation, some heuristics are used to further decrease the number of regions by eliminating small regions and merging weakly contrasted regions with adjacent regions. The edge pels found in the segmentation step are grouped into contours and each contour can be coded as a piece-wise approximation using line segments, or circle segments, or without any approximation. The texture in each region can be assumed to be relatively smooth with no discontinuities so that two-dimensional polynomial function is fit over the region. It has been reported that the first- or second-order polynomial is adequate if the granularity removed is added back in the form of

random noise. The variance of the noise is controlled by the mean-square error between the original and the reconstructed image without the noise.

Although a very high compression ratio is reported for this coding scheme, on the order of 50 to 1, the reconstructed image quality is quite poor. Even with the random noise, it still has the "painted by the number" look and has many of the important low-contrast edges missing. Also some of the seemingly smooth edges are distorted and in general poorly reproduced. It seems that most of the problems can be rectified in various ways. The segmentation algorithm should produce contours that line up better along the perceptual edges in the image. Incorporation of the gradient information should help. The contours along high-contrast edges should be reproduced more faithfully, whereas that of low contrast may be coded with less fidelity. The junction points where three or more contours converge should also be reproduced well since they correspond to the corner points of adjacent regions. The fidelity of the texture reproduction needs to be improved by increasing the polynomial order in large regions or perhaps by resorting to different coding methods such as transform coding or predictive coding. The *micro-texture* added in the form of random noise can be better characterized by including directionality of the variance and higher-order statistics.

2.3 Contour-Based Motion Estimation

The motion estimation can be broadly classified as intensity-based and token-matching schemes [8]. Psychovisual evidence supports that both schemes may be present in human vision. The short-range motion based on intensity takes place early on in the visual chain and the long-range motion based on matching occur at higher levels in the brain. The contour-based scheme can be classified as a token-matching scheme where part or whole of the contour is used as a token.

The Hough transform was originally developed to detect the presence or absence of a straight line parameterized by $y=ax+b$. The transform maps a line in the image onto a point in the parameter space. A clustering in the parameter space indicates a presence of a line. Later the Hough transform was generalized to detect curves that can be described by analytic functions. Darmon [9] has shown a way to linearize the equations of the contour and recursively estimate the parameters when the a priori estimate is available. The estimate is available when processing a sequence of images except at the beginning, where the initialization is performed either manually or by applying the Hough method itself on the first image. This technique seems to be limited in its use to simple scenes with a handful of contours and is not very attractive for our purpose.

Davis has proposed a contour-based technique where the motion is estimated at the corner points of the contour, then propagated along the rest of the contour [10]. Based on the observation that the motion can be computed unambiguously at corners from measurements made only at or near the corner points, two approaches are taken. A structural approach measures the location, corner shape (angle of two line segments that form the corner), and corner contrast. A least-squares approach that resembles a gradient method is applied around the corner. Rotation as well as rigid translation can be accommodated this way. The velocity obtained at the corner is then propagated as the normal component and the tangential component. An error occurs

if a propagation from one corner does not agree with a propagation from another corner at the midpoint between two corners. This error has to be redistributed to avoid the discontinuity.

A more fundamental and theoretical analysis of motion along contours was developed by A.I researchers for applications in computer vision [11,12]. A local measurement of motion can only compute one component of the velocity, and in case of a contour only the component normal to the contour can be directly computed. An integration of local measurements along the contour is necessary to compute the full velocity field. However, the integration can be ambiguous, as pointed out by Hildreth. Based on human perception, which prefers the least variation in velocity, a smoothness constraint is imposed. It is found that a mathematically unique velocity can be found by minimizing the variation of the velocity function along the contour in the form of a functional [11].

$$\int \left| \frac{\partial V(s)}{\partial s} \right|^2 ds$$

$V(s)$ is the velocity vector along the contour parameterized by the arc length s. Yuille [13] has shown mathematically that a token-matching scheme that matches tangent vectors along the contour gives the equivalent result that could be gotten by imposing the smoothness constraint to the above equation.

2.4 Texture

Once we have a set of contours representing the boundaries of segmented regions, we need to describe the texture in each region so that the original image can be reconstructed. A well segmented region should exhibit a homogeneous texture that can be efficiently represented by a small number of parameters. A good texture model is essential in order to describe the region compactly for coding purposes. Unfortunately, there are relatively few texture models that are aimed directly at coding segmented regions; most research efforts are oriented toward a pattern recognition problem or a synthesis problem for computer graphics.

Most coding-oriented approaches so far have adopted very simplistic methods: Kocher [17] has assigned a single gray level value as the average brightness of the region. Kunt [7] refined it by fitting a two-dimensional polynomial function of a small order (0 to 2). A Gaussian random noise was added as a "microtexture" to minimize the cartoon-like look. Lemay [57] coded only those pixels along the contour and ignored the variation inside the region. Schreiber [5] suggested the texture be modeled by a low-pass random noise with vertical and horizontal bandwidths plus power. Anderson [58] coded texture by Fourier spectrum magnitude thresholded and quantized, and added random phase to obtain a compression ratio of over 10. Margos [59] has applied a linear predictive analysis on arbitrarily shaped regions, but their goal was texture discrimination based on LPC distance measures. For our purposes, a more refined model needs to be developed in order to produce a natural-looking picture while maintaining a good coding efficiency.

A precise definition of texture is quite difficult and often ambiguous since the notion of texture is largely subjective. To most people, texture means a repetitive pattern of some sort, not necessarily regularly spaced. This description may apply well for a picture of brick wall or wire braid or closely woven fabric where the texture primitives that make up the texture are relatively easy to

identify. However, other materials such as grass, sand, or concrete wall have no discernible pattern and the texture primitive is less obvious. The former fits into the structural model in which the placement of well-defined primitives is governed by a generation rule. The latter, on the other hand, fits into the statistical model in which the distribution and relation of gray levels are described instead, because of the lack of apparent texture primitives.

It may be tempting to assume that the structural model is suitable for generating macrotextures and the statistical model is adequate for microtextures. In fact, this assumption is true in many cases and the division into two models has been widely accepted. For example, Zucker [60] describe a structural model that views the real texture as a distorted version of the ideal texture. The ideal texture is represented on a regular tesselation of the plane and this tesselation can be distorted to approximate the real texture. However, the distortion process may involve random parameters. An example of a statistical model can be found in [61] which explores the Markov random field model. Their model assumes a two-dimensional Markov process with directional clustering property (for extension of Markov model to color texture, see [62].) However, this division into two models cannot be too rigid since a statistical model can describe pattern-like textures and vice versa. Haralick, for example, discusses hybrid approaches that apply statistical techniques to the structural primitives, along with an excellent survey of both models in [63]. For a typical discussion on texture representation oriented toward machine vision, see [64].

So far, current understanding of texture is not adequate for our needs for the following reason: First, the appropriate texture model we need to adopt is strongly dependent upon the segmentation algorithm. We have to know whether the segmented region will contain macrotextures or only microtextures. Some segmentation methods even use texture models themselves [65]. It is anticipated that gradient-based segmentation, which is one of the strong candidates for the type of coding we envision, will produce only

microtextured regions since the macrotextured area will be further segmented into smaller regions. However, it is possible to identify a group of similarly shaped small regions and aggregate them into a macrotextured region. In that case the small region can serve as the texture primitive for a structural model. It is interesting to consider a generalization of a structural model, in which the whole image is viewed as a textured region. Then the image can be hierarchically represented in a tree structure where each branch corresponds to a textured region which may be recursively segmented further to form a subtree. The leaves of the tree will be the primitives of a structural model.

The second difficulty stems from the fact we are trying to code sequences of images. Suppose we have identified two (or more) corresponding regions from two (or more) successive frames of an image sequence. If we adopt a statistical texture model, the variation of the texture over time within the region would be very noticeable, even though each frame by itself were quite acceptable subjectively. If we adopt a structural model, this randomness may disappear, but many microtextured regions cannot be efficiently coded because the size of the texture primitives will be on the order of a few pixels. The dilemma is that we want to reconstruct the texture on a point-by-point basis to some extent even for microtextures when coding image sequences, in order to avoid producing temporal noise. That is, we need a three-dimensional model for texture that incorporates the time axis. Although there does not seem to be any work on 3-D texture models, it is possible to extend a 2-D model, either statistical or structural. In light of the cylinder model for the moving region, however, it may simplify the matter by discarding the texture model and adopting a straightforward coding, perhaps a scheme using two-channel coding spatially and predictive coding temporally within the region. It is clear, at least, that the coding efficiency will depend very much on how the texture is handled; therefore a lot of attention must be given to texture coding.

2.5 Segmentation

The segmentation problem arises in a wide variety of applications ranging from locating mitochondria of a liver cell from electron micrographs [31] to identifying machine parts from industrial images [27]. It is an essential first step in most machine vision systems because higher level information about objects such as structure and orientation can be inferred from the segmented image. The importance of segmentation for processing images is very much apparent in the amount of literature available on the subject, especially in the field of pattern recognition and scene analysis. Unfortunately the large amount of research effort in the literature also indicates that segmentation is an inherently difficult problem that is far from solved. The main problem seems to be the noise, either introduced during the imaging process or something inherent in the image such as surface texture. Although one can find an algorithm that works on a particular class of images taken in a highly controlled environment, there is no one algorithm that works well on natural images such as outdoor scenes. Because of the difficulty, widely different approaches have been suggested, not to mention minor variations and improvements on each approach.

The simplest idea for segmentation is intensity thresholding, which requires no neighborhood interaction in decision making. It performs poorly in most cases, except perhaps for line art and typographical characters, which are essentially binary images. However, thresholding forms a basis for a number of approaches, such as histogram-based and clustering techniques. A histogram-based approach [14] recursively splits the image into regions based on the histogram of each region. Usually two or more peaks in the intensity histogram are identified at each iteration. The value near the valley between the peaks is then used to split the current region into two (or more) by thresholding. Some form of noise elimination is necessary to remove small holes in the region. Chromaticity can be used to augment the intensity for selecting the threshold and the resulting segmentation can be quite convincing even on natural scenes [15].

The clustering technique is closely related to the histogram methods where the threshold is selected in the *feature space*. The feature space is formed from various measurements from the intensity such as brightness and texture. Coleman and Andrews [16] have used a dozen measurements based on the brightness as well as the magnitude and phase of the Sobel edge operator, each of them computed using different window sizes. A *mode filter* similar to a median filter was applied to each measure to avoid producing small fragments. Here the Sobel operator output represents some measure of texture, so other edge operators could have been used instead. The results were then rotated to decorrelate the features and a feature reduction is performed to retain several features for good clustering. Then the actual segmentation process becomes just a multi-dimensional extension of the histogram approach.

The relaxation methods and region-growing methods are also closely related. The region-growing method finds a small group of pixels with similar intensity and then merges adjacent groups to form regions [17]. Therefore a "hard decision" is made at each merging step. In a relaxation method, on the other hand, each pixel is explicitly assigned a probability of belonging to a particular class. Then this probability is iteratively adjusted on the basis of the adjacent pixels. It is only after the final iteration that each pixel is classified as belonging to a region of the maximum probability. Eklundh [18] has applied the relaxation method to color images. Hanson [19], on the other hand, performed relaxation based on the probabilities mapped from the feature clusters. The idea was to improve the spatial clustering of the clustering algorithm. Rutkowski [20] applied relaxation techniques to a slightly different problem where probabilities are assigned between boundary segments represented by a directed graph. In general, region growing and relaxation methods do a good job segmenting gross shapes, but at the expense of obscuring fine detail.

The segmentation approach described so far either progressively splits large regions into small ones or progressively merges regions

based on the similarity of the pixels in the region. Another popular approach is based on edge detection, where the boundary of the region is extracted as edge points. First a local edge operator such as Sobel, Roberts, Kirsh, Prewitt, etc., is applied to find edge points and assign edge strengths. For a survey on edge detection see [21] and [22]. More discussion on edge detection can be found in [23,24]. Once the edge points are found they are linked according to some heuristics to form a closed outline or boundary of the region. One of the simple ways to group edge points into edges is to threshold the edge strength and apply a thinning operation [25] to reduce the edge thickness to a single pixel. However the threshold method reveals a problem common to most edge based approaches: depending on the threshold value, gaps appear between edge segments which must be bridged in order to get a closed region. To solve this problem, Basseville [26] has presented a recursive edge-following scheme based on a Kalman filter using a noisy straight line as the model. Perkins [27] proposed an expansion-contraction technique where edge points are first expanded to bridge the gaps and then contracted back using a connectivity analysis similar to the thinning operation. Prager [28], on the other hand, applied a relaxation scheme to edge points to improve the edge grouping.

Other segmentation schemes include a texture-based approach used by Derbin [29] where the texture was modeled by a Gibbs distribution and a dynamic programming approach was taken for segmentation. Pal [30] has introduce a notion of fuzzy sets to enhance the contrast between regions before edge detection. Gritton [31] describes a bead chain algorithm for locating a boundary from a known initial approximation. For general discussion on image models for segmentation, see [32,33].

2.5.1 Segmentation Experiments

One important property a segmentation scheme must have in order to be useful in our application is that the segment boundary should coincide with perceptual edges as much as possible. This requirement favors the edge-based approach. Another requirement is

the accuracy of the segmentation in terms of the edge location. For a good reconstruction from contours, the edge location needs sub-pixel accuracy. This implies some form of interpolation when extracting edge points. Unfortunately, the bulk of segmentation schemes were developed as a preprocessing step for a scene analysis and understanding system. Consequently, the edge locations are quantized to a pixel spacing, and *edge pixels* are often used to mark the boundary between regions, making it ambiguous to decide to which region the edge pixels belong.

The following two experiments are aimed at satisfying the above two requirements. The first is a simple-minded thresholding technique in which a bilinear interpolation is performed for sub-pixel accuracy. The contours generated this way are called iso-luminance contours because the luminance value is constant along the contour. A careful selection of the threshold value can produce an excellent segmentation for a simple image as in Fig. 2.1, but results in a somewhat ambiguous segmentation on more natural image as in Fig. 2.2. However the main advantage of this scheme is that it always produces closed contours (except at the image boundary) and the computation is very simple. Perhaps a histogram approach described above can be employed to select the threshold adaptively. Fig. 2.3 shows the result of using four threshold values. A suitable histogram method may be able to eliminate redundant contours and leave only the essential ones.

The second experiment is based on edge detection after a curve fitting over the image. A two-dimensional cubic B-spline is fit over a 4-by-4 window so that gradients can be computed from the spline coefficients. The cubic B-spline was chosen because it has continuous first and second derivatives and also because it is a good interpolation function, comparable to a Gaussian function. Fig. 2.4 compares a gradient magnitude picture produced this way against a conventional edge operator. The curve-fitting approach can detect edges along a very thin line whereas the conventional edge detector obscures the line. Once we have the image intensity function in analytic form, a

two-dimensional polynomial in this case, we can get accurate gradient by computing the directional derivatives. Then we can trace out the contour of maximum gradient or gradient peaks. Conceptually, the gradient peak corresponds to the zero-crossing contours of the second derivative so that we should be able to get a piecewise polynomial description of the gradient contour by solving for the roots. However, solving for roots in this case is not trivial as it involves solving polynomials of two variables. In practice, we can compute gradients at several places around a pixel and link these points according to the magnitude and angle of the gradient.

One simple method is to link two points whose dot product of the gradients is maximized. This insures edge points of similar orientation get linked even when the gradient magnitude is low. The contour tracing stops when the dot product becomes smaller than a threshold, the contour runs into another contour thereby producing a junction, or the contour closes itself. Fig. 2.5 shows a preliminary result of the gradient contours produced in this manner. Although most of the noticeable edges were detected, many contours failed to link properly and were left open. A close examination indicates that saddle points are responsible for the majority of these cases. A saddle point is where more than two contours can meet and produce a junction point. Fig. 2.6 shows two saddle points where the algorithm may fail. To make this algorithm useful we need to find a way to treat these saddle points separately. If open contours still remain, then other contour following or gap filling procedure will be needed. In addition, some of the small noisy contours must be removed by examining the length and edge strength.

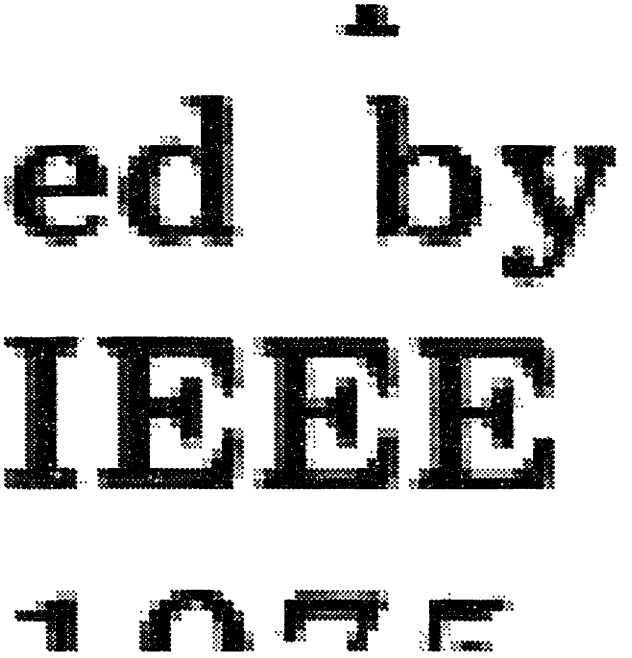


Figure 2.1(a)
Text image, 64 by 64

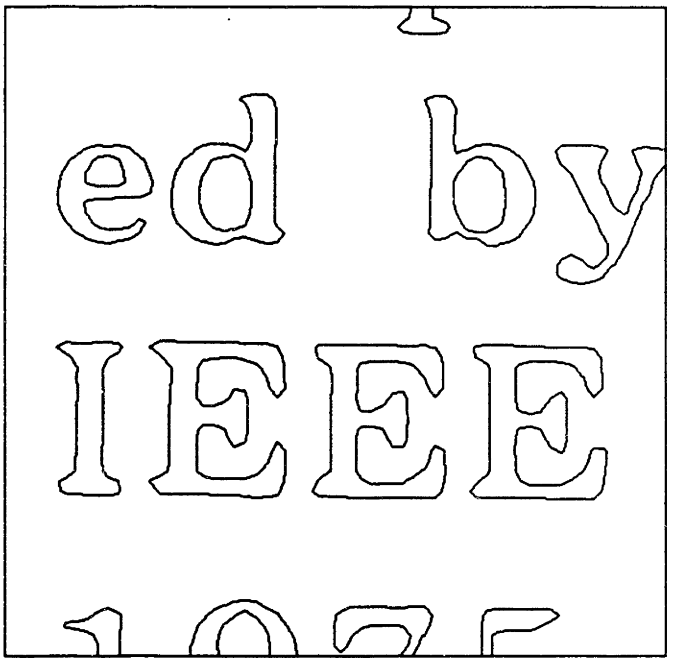


Figure 2.1(b)
Text segmented by thresholding

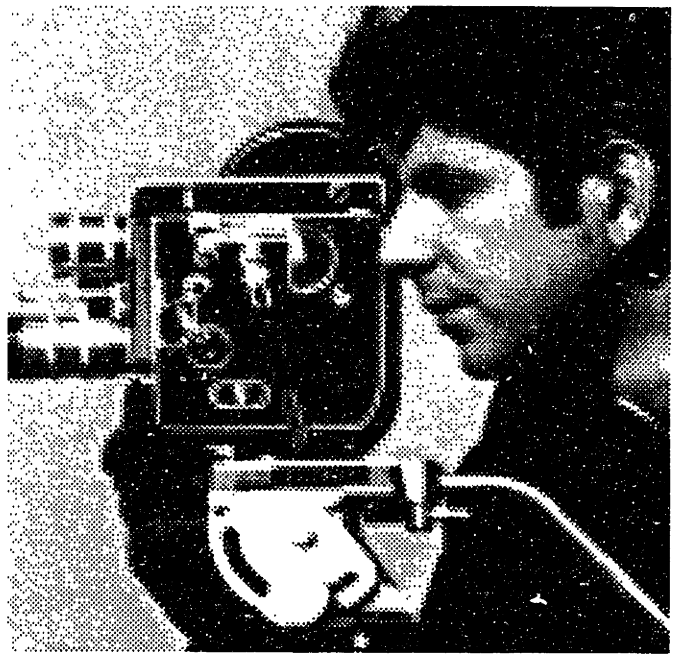


Figure 2.2(a)
Camera man (cman) image, 128 by 128

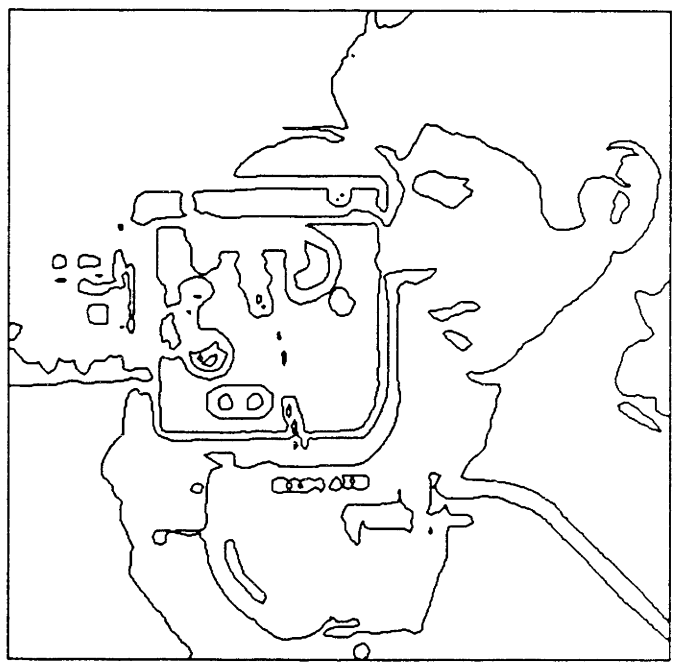


Figure 2.2(b)
Cman segmented by thresholding

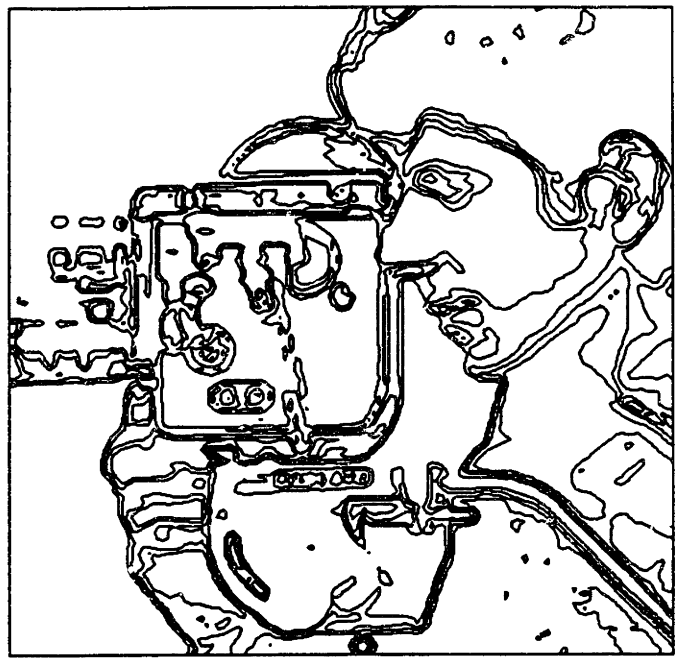


Figure 2.3
Multiple thresholds of cman.

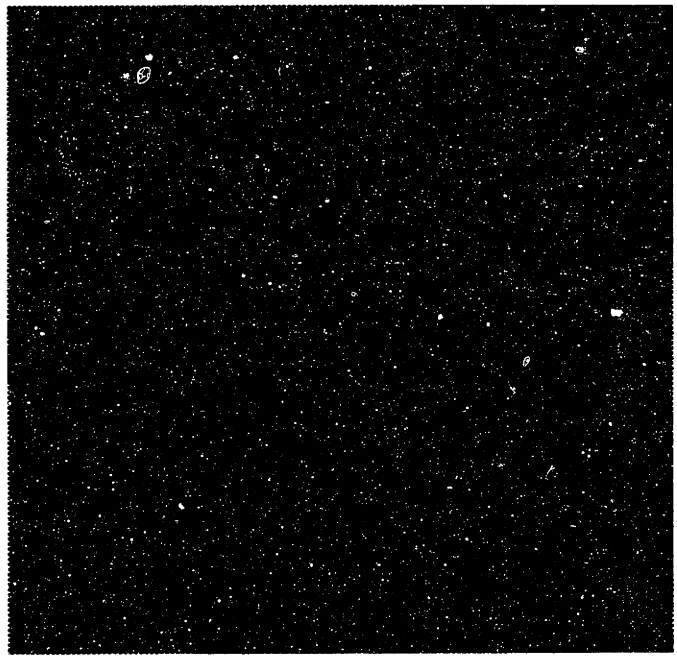


Figure 2.4(a)
Sobel magnitude of cman

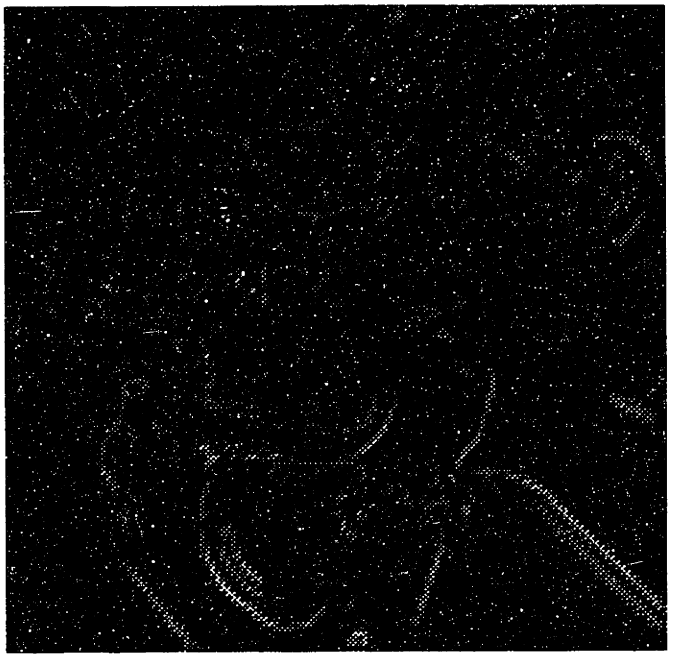


Figure 2.4(b)
Gradient computed by sum of the differences
at four nearest neighbor pixels

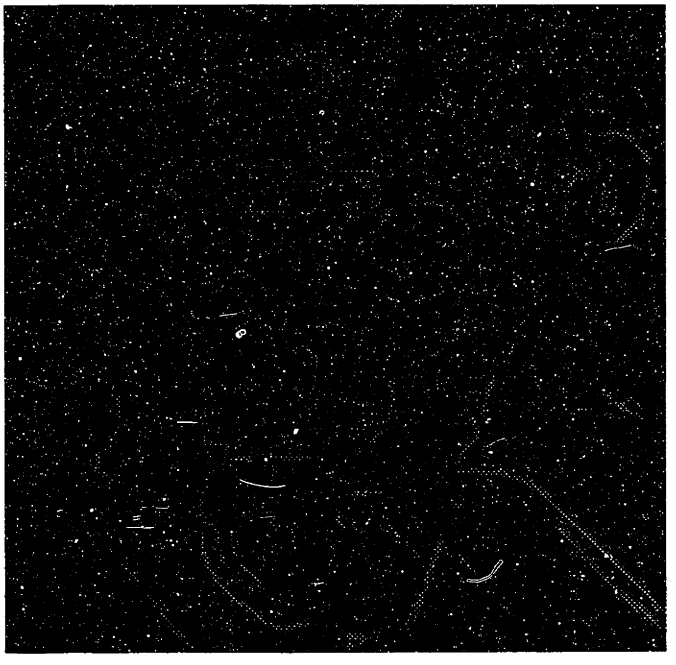


Figure 2.4(c)
Gradient magnitude from bi-cubic B spline fitting

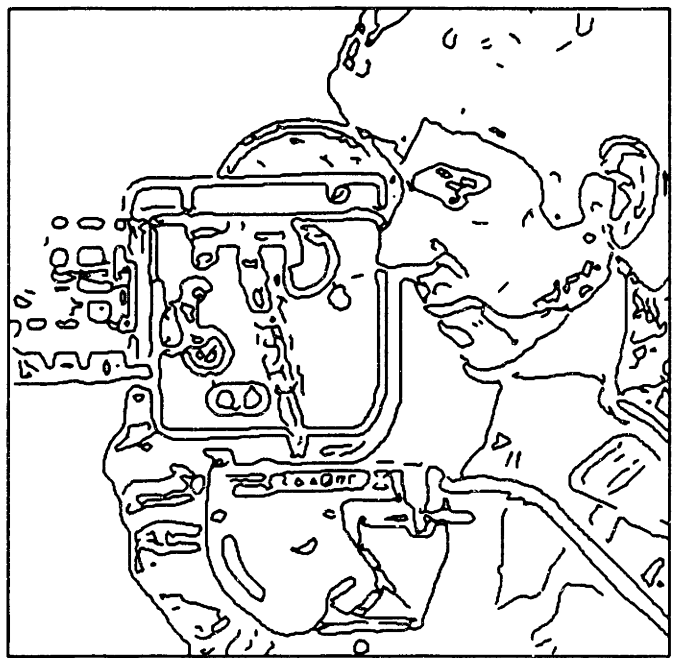


Figure 2.5
Gradient contours from figure 2.4(c).

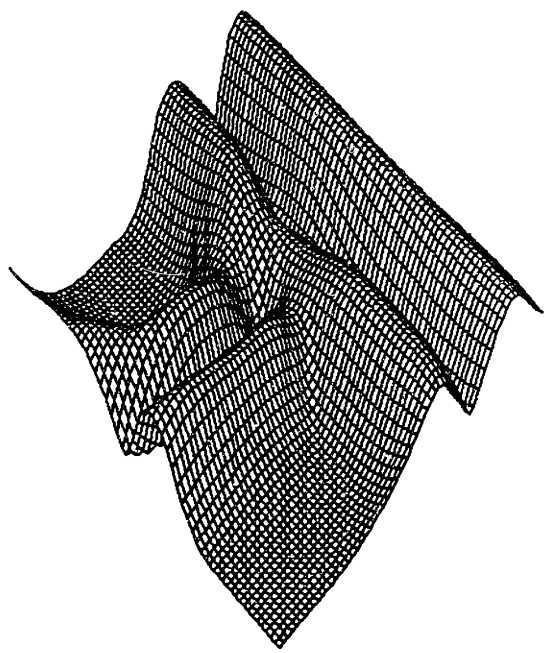


Figure 2.6(a)
Perspective view of two saddle points in gradient magnitude.

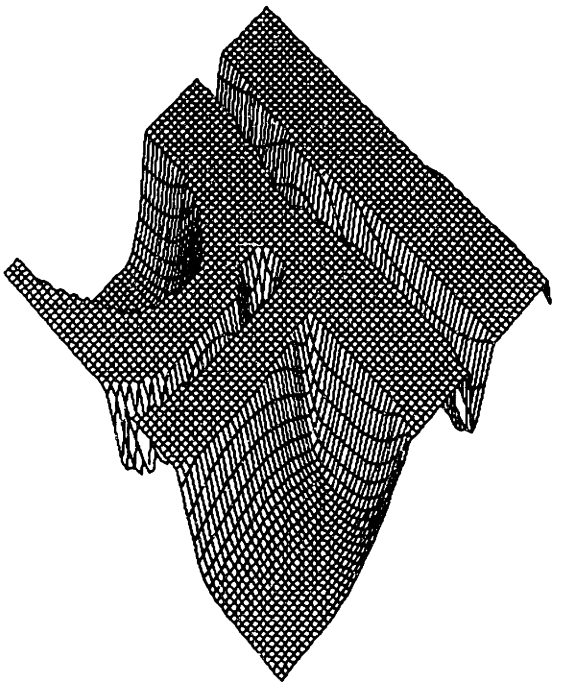


Figure 2.6(b)
Perspective view of the thresholded gradient magnitude (from figure 2.6(a)) to show the two junction points.

2.6 Contour Coding

After the image has been segmented into regions, we need an efficient way to represent the resulting contour and texture. Representing the contour is of primary concern since most of the shape information is embedded in the contour. Even if we can obtain the contour to an arbitrary precision, it is desirable to approximate the contour by a finite number of samples for storage, transmission, or manipulation. However, a representation scheme efficient for storage and transmission may not necessarily be the most convenient form for manipulation. Investigation of different contour representations and manipulation techniques is thus an important research topic.

A piece-wise approximation is by far the most popular approach taken in the literature for representing an arbitrarily shaped contour. In encoding typographical characters, the outline of a character is usually divided into a series of curve segments, where each segment is actually a part of a mathematical function such as a circle or a spiral. Fah [34] has demonstrated a character-coding technique that combines both circle segments and line segments. Although curve-fitting methods work well for smooth and predictable outlines of printing characters, they are not so suitable for arbitrary contours such as a shoreline in a map, outline of a chromosome, etc., because the lengths of the curve segments tend to be very short.

For an arbitrary contour, a polygonal approximation is the accepted canonical form in image processing and computer graphics. A polygonal approximation is equivalent to a piecewise linear approximation using straight lines. The goal is to represent the contour using the minimum number of vertices (line segments) while satisfying a given fit criterion. The fit criterion can be maximum absolute deviation, mean-square error, or absolute area between the contour and the polygonal approximation, etc. For example, Ramer [35] presented a computationally efficient iterative algorithm for finding the polygonal approximation. It is based on iteratively splitting the curve segments at the point of maximum deviation.

Recently Dunham [36] proposed an optimum iterative procedure using the uniform l_∞ norm and made a performance comparison against other algorithms, including Ramer's. His conclusion suggests that all algorithms perform similarly well for a small error criterion. Although his optimum approach always produces the solution with the least number of vertices, the computation time increases for a large error criterion.

Although the polygonal approximation quantizes the contour into a finite number of samples (vertices), it does not suggest a way to quantize the sample coordinates. Another successful and widely used method for coding contours is the chain code, originally developed by Freeman [37]. The chain code can be thought of as a two-dimensional extension of the delta modulation technique, in which the position of the next vertex from the current vertex is encoded differentially as up, down, left, and right. The unit of movement is one grid spacing. In addition to these four directions, four diagonal directions can be added to improve the performance. Different chain codes are obtained depending on how the grid points are chosen: Square quantization defines a non-overlapping square area centered around the grid point and assigns the grid point to the chain if the contour goes through the square; Circular quantization defines a circle instead of a square whose diameter is same as the grid spacing; Grid-intersect quantization picks the nearest grid point from the point where the contour intersects the grid line. The square quantization does not use diagonal directions and therefore requires only 2 bits to encode, as opposed to 3 bits for the other two. Koplowitz [38] has generalized these chain codes by the p-Minkowski metric quantization for which the above three methods become special cases.

Many simple manipulations and extraction of contour parameters are possible with the chain code and are illustrated in [39] and [40]. A procedure for scaling of a chain coded contour without resampling was reported in [41]. The quantization error analysis and coding efficiency of various chain codes can be found in [42] and [43].

A practical curve generation technique is presented in [44]. Other contour representation methods can be found in [45] and [46].

2.7 Shape Analysis

It is desirable to manipulate the contours in various ways while preserving their general shape. Rotation, translation, and scaling are shape-preserving transformations useful for shape analysis. For example, a contour extracted from the image can be rotated, translated, and scaled in size to match stored library contours for shape classification or object recognition. However, determining these three transformation parameters from a polygonal or chain code representation is not trivial. Therefore it is advantageous to have a representation that retains only the shape information so that the similarity measure between two contours can be made easily.

There are several different techniques for transforming a contour such that the resulting representation is invariant to rotation, translation, and scaling. Zahn [48] used the Fourier descriptor of Cosgriff [49] for the analysis and synthesis of closed contours. He used the coefficients of the Fourier series expansion of the angle versus arc-length function so that all redundant parameters are removed. Granlund on the other hand computed the Fourier transform of the complex representation of the contour in which the real axis is the x-coordinate and the imaginary axis is the y-coordinate [50]. Then the properties of the Fourier transform were used to factor out the rotation, translation, and scale parameters. Dubois [51] has proposed an autoregressive model from a representation that divides the contour into a number of radius vectors equally spaced in angle. The approximation is then given by the sequence of radii from the centroid to the boundary. A scale-space approach was taken by Mokhtarian [52] in which the zeros of curvature of the contour is plotted at varying levels of detail. A Gaussian kernel of varying σ was used to control the scale of the detail. This too produces a representation which is independent of the three parameters. For other scale-space approaches see [53].

The above techniques were mostly developed for pattern recognition problems where a contour is compared against a known

shape for classification. Granlund applied the Fourier descriptor for recognizing hand-printed characters. Mokhtarian used the scale-space for registering Landsat satellite images to a known map. Richard [54] and Wallace [55] described the use of the Fourier descriptor for aircraft identification. In this thesis, however, we are interested in utilizing these techniques for matching two unknown contours for the purpose of motion estimation. The discussion in the following chapter on contour transform is based on independent research but it was later realized that similar work was done by others. It closely resembles the developments by Granlund [50], Richard [54], Persoon [56], and later by Wallace [55]. The research is then further extended to develop a robust contour motion interpolation technique. In Chapter 4, we will develop new techniques for matching *open* contours which cannot be treated otherwise using the transform techniques.

CHAPTER 3. CLOSED CONTOURS

In this chapter we assume that the contour is *closed*. The contour transform we are about to describe is based on the fact that the contour can be represented by a periodic signal, which requires the contour to be closed. The transform technique is shown to provide very convenient properties for separating the shape parameters from the orientation information, i.e. translation, rotation, and magnification. These properties lead to the development of efficient matching and interpolation techniques.

3.1 Contour in Complex Plane

Pick an arbitrary point on the contour as the origin and use the parameterization s as the distance from the origin along the contour as the contour is traversed counter-clockwise. Then the set $\{x(s),y(s)\}$ completely describes the contour, where $x(s)$ is the x coordinate and $y(s)$ is the y coordinate of a contour point at s. Now let $c(s)$ be a complex signal that represents the contour such that the real part is $x(s)$ and the imaginary part is $y(s)$.

$$c(s) = x(s) + jy(s) \tag{3.1a}$$

Alternatively, $c(s)$ can be put into a polar form.

$$c(s) = r(s)\, e^{j\theta(s)} \tag{3.1b}$$

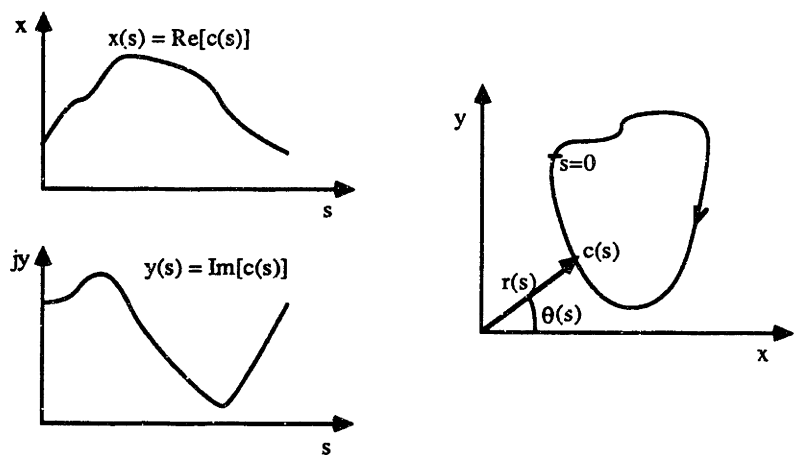


Figure 3.1
Complex representation of contour.

This represents the contour by a *one-dimensional* complex signal. An important observation we make here is that if the contour is closed, $c(s)$ is truly a periodic signal and can be subjected to a Fourier analysis. If the contour is continuous in s, from the nature of periodic signals, we get a Fourier series. If the contour is discrete, we can use Discrete Fourier Transform (DFT).

Suppose a contour is given by a polygon whose N vertices are equally spaced in arc length. Then the contour can be represented by a complex periodic signal as above with a period of N.

$$c(n) = x(n) + jy(n) \tag{3.2a}$$

$$c(n + iN) = c(n) \qquad i = \ldots,-2,-1,0,1,2,\ldots \tag{3.2b}$$

And in polar form,

$$c(n) = r(n)\, e^{j\theta(n)} \tag{3.2c}$$

3.1.1 Circular Transform

We now define the circular transform as the Fourier transform of the contour signal $c(n)$. The resulting transform pairs are no different from the ordinary DFT pairs:

$$C(k) = \sum_{n=0}^{N-1} c(n)\, e^{-j\frac{2\pi}{N}kn} \tag{3.3a}$$

$$c(n) = \frac{1}{N}\sum_{k=0}^{N-1} C(k)\, e^{j\frac{2\pi}{N}kn} \tag{3.3b}$$

The transform coefficients $C(k)$ can be represented more naturally in polar coordinates:

$$C(k) = m(k)\, e^{j\phi(k)} \tag{3.4a}$$

where

$$m(k) = |\, C(k)\,| \tag{3.4b}$$

$$\phi(k) = \arg[C(k)] \qquad -\pi \le \phi(k) \le \pi \tag{3.4c}$$

From this we can see that the contour is represented by a linear combination of circles of different radius, frequency, and phase. At any frequency k, $m(k)$ is the radius and $\phi(k)$ is the phase (starting point) of a circle that spins k revolutions per period (the period is N). Therefore at zero frequency (at DC), $m(0)$ and $\phi(k)$ are the polar

coordinate of the center of the contour (i.e. the centroid). At k=1, the contour traverses a circle counter-clockwise exactly once over a period. At k=2, it traverses at twice the speed, making 2 circles over a period, and so on. The circles traverse clockwise for negative frequencies. When all are combined, they can traverse an arbitrarily shaped contour. Hence the basis functions of this transform are circles instead of sinusoids. For this reason this transform was originally named *circular transform.* However, this is identical to the Fourier descriptor used by Granlund. Since there are other kinds of Fourier descriptors, we shall refer to this specific procedure as either circular transform or, when appropriate, *contour transform.*

3.1.2 Center of Contour

As indicated above, we define the center of a contour as the average of the contour points. This can be thought of as the center of gravity of the contour when all the contour points are given identical weights. The center of contour is

$$C_c = x_c + jy_c \tag{3.5a}$$

where

$$\begin{aligned} x_c &= \frac{1}{N}\sum_{n=0}^{N-1} x(n) = \frac{1}{N}\sum_{n=0}^{N-1} \mathrm{Re}[c(n)] \\ &= \frac{1}{N}\mathrm{Re}\left[\sum_{n=0}^{N-1} c(n)\right] = \frac{1}{N}\mathrm{Re}[C(0)] \end{aligned} \tag{3.5b}$$

and,

$$y_c = \frac{1}{N}\sum_{n=0}^{N-1} y(n) = \frac{1}{N}\mathrm{Im}[C(0)] \tag{3.5c}$$

or simply

$$C_c = \frac{1}{N}C(0) \tag{3.5d}$$

Obviously, the DC term of the transform corresponds to the center of the contour -- i.e. displacement from the origin.

3.1.3 Translation

Let d be the displacement vector (translation by x_d and y_d).

$$d = x_d + jy_d \tag{3.6}$$

Then the translated contour is

$$c'(n) = c(n) + d \tag{3.7a}$$

whose transform is

$$\begin{aligned} C'(k) &= \sum_{n=0}^{N-1} (c(n) + d)\, e^{-j\frac{2\pi}{N}kn} \\ &= \sum_{n=0}^{N-1} c(n)\, e^{-j\frac{2\pi}{N}kn} + d\sum_{n=0}^{N-1} e^{-j\frac{2\pi}{N}kn} \\ &= C(k) + dN\delta(k) \end{aligned} \tag{3.7b}$$

Therefore the translation only affects the DC term at $k=0$. The following properties assume that the contour is already translated to the origin for simplicity -- i.e. $C(0)=0$.

3.1.4 Magnification (zooming)

With $C(0)=0$, assume we would like to scale the contour by a magnification factor of f. Then the scaled contour is

$$c'(n) = f\, c(n) \tag{3.8a}$$

whose transform is

$$C'(k) = f\, C(k) = f\, m(k)\, e^{j\phi(k)} = m'(k)\, e^{j\phi'(k)} \tag{3.8b}$$

so that

$$m'(k) = f\, m(k) \tag{3.8c}$$

$$\phi'(k) = \phi(k) \tag{3.8d}$$

So, magnification involves scaling the magnitude of the transform.

3.1.5 Rotation

Again with $C(0)=0$, let us rotate the contour by an angle θ_r counter-clockwise. This is easily accomplished in polar coordinate.

$$c'(n) = r'(n)\, e^{j\theta'(n)} = r(n)\, e^{j(\theta(n)+\theta_r)} \tag{3.9a}$$

$$\begin{aligned} C'(k) &= \sum_{n=0}^{N-1} r(n)\, e^{j(\theta(n)+\theta_r)}\, e^{-j\frac{2\pi}{N}kn} \\ &= \left[\sum_{n=0}^{N-1} r(n)\, e^{j\theta(n)}\, e^{-j\frac{2\pi}{N}kn} \right] e^{j\theta_r} \\ &= C(k)\, e^{j\theta_r} = m(k)\, e^{j(\phi(k)+\theta_r)} \end{aligned} \tag{3.9b}$$

so that

$$m'(k) = m(k) \tag{3.9c}$$

$$\phi'(k) = \phi(k) + \theta_r \tag{3.9d}$$

Therefore the rotation only adds a constant phase to the phase of the transform.

3.1.6 Circular Shift

Circularly shifting the contour samples in one direction or another does not change the shape or orientation of the contour. It only redefines the origin of the parameterization. This is significant when the contouring algorithm cannot supply a consistent initial point, which is usually the case. As expected, the shift operation adds a linear phase to the transform. Assume a shift by l points:

$$c'(n) = c(n-l) \tag{3.10a}$$

$$C'(k) = \sum_{n=0}^{N-1} c(n-l)\, e^{-j\frac{2\pi}{N}kn} \tag{3.10b}$$

Let $n' = n - l$. Then,

$$C'(k) = \sum_{n=0}^{N-1} c(n')\, e^{-j\frac{2\pi}{N}k(n'+l)}$$

$$= C(k)\, e^{-j\frac{2\pi}{N}kl} = m(k)\, e^{j\left(\phi(k) - \frac{2\pi l}{N}k\right)} \tag{3.10c}$$

so,

$$m'(k) = m(k) \tag{3.10d}$$

$$\phi'(k) = \phi(k) - \frac{2\pi l}{N}k \tag{3.10e}$$

The significance of the above properties that relate to the geometric transformation is that translation, zoom and rotation can be achieved independently of one another in the transform domain. A new contour $c'(n)$ that has been translated by $d = x_d + jy_d$, scaled by f, rotated by θ_r, and circularly shifted by l points can be expressed in the transform domain as:

$$C'(k) = f\, e^{j\left(\theta_r - \frac{2\pi l}{N}k\right)} C(k) + dN\delta(k) \tag{3.11}$$

3.2 Motion from Contours -- Contour Matching

Using the circular transform properties developed above, we can easily extract the parameters of the geometric transformation necessary to orient a contour to best match another contour. A suitable similarity measure can be used to compare two contours to establish a correspondence. Once a match is made, we can readily use the resulting motion information for a temporal interpolation.

3.2.1 Mean Square Error between Contours

Since the circular transform is a unique decomposition of the contour signal into an orthonormal basis (the basis functions being circles in the complex plane), the Euclidean distance in transform space can be used as the similarity measure. We define the mean square error between the two contours i and j as follows:

$$mse_{ij} = \frac{1}{N}\sum_{k=1}^{N-1} | \bar{C}_i(k) - \bar{C}_j(k) |^2 \qquad (3.12)$$

where N is the transform length, and $\bar{C}_i(k)$ and $\bar{C}_j(k)$ are the normalized transform coefficients of $C_i(k)$ and $C_j(k)$ such that the two contours are of same scale and orientation. The normalization will be discussed in detail in the following section.

Notice that the DC term at $k=0$ is excluded to avoid the translational component. Having excluded the translational component from the MSE measurement, we are left with the problem of factoring out the magnification and rotational components by normalizing the magnitude and phase of the circular transform. Both components are relative parameters between similarly shaped contours: one can only speak of the magnification factor and the rotation angle of a contour relative to another contour. Thus, if C_i is a contour in frame 1 and we want to find one contour among many in frame 2 that best matches C_i, we would let $\bar{C}_i(k) = C_i(k)$ and transform each contour in frame 2 (and substitute it in for $\bar{C}_j(k)$ to minimize Eq. (3.12). It is quite possible to solve for the magnification and rotation values for each contour pair

that minimize the MSE between two contours provided N is greater than 2. However this would be a computationally very expensive approach if the solution has to be found for every permutation pairs of contours in frame 1 and 2. To solve this problem we propose a simpler scheme that normalizes the contours with respect to a universal reference such that a reasonable approximation to the MSE is obtained without much computation.

3.2.2 Normalization of Magnitude and Phase

In Eq. (3.12) we were able to factor out the translation term by effectively translating each contour to the origin, thus using the origin as the reference. Similarly we can factor out the magnification and rotation terms if we have a consistent reference. To do this we turn to the magnification and rotation properties of the circular transform (magnification by f and rotation by θ_r):

$$m'(k) = f\, m(k) \tag{3.13a}$$

$$\phi'(k) = \phi(k) + \theta_r \tag{3.13b}$$

Unfortunately there is an additional parameter θ_s (slope of the linear phase due to circular shift of the contour points) to deal with when the origin of the contour is unknown. This is another relative parameter. θ_s is undesirable since it does not influence the shape or the orientation of the contour. Nevertheless, it is essential to completely describe the contour. When combined with the rotation term, we get

$$m'(k) = f\, m(k) \qquad 1 \le |k| \le \frac{N-1}{2} \tag{3.14a}$$

$$\phi'(k) = \phi(k) + \theta_r + \theta_s k \tag{3.14b}$$

or

$$C'(k) = m'(k)\, e^{j\phi'(k)} = f\, e^{j(\theta_r + \theta_s k)} C(k) \tag{3.14c}$$

where $\theta_s = -2\pi l/N$ when the shift is by l sample points (see the circular shift property). Now we define the normalized magnitude and phase as follows:

$$\bar{m}(k) = \frac{m(k)}{\bar{f}} \qquad 1 \le |k| \le \frac{N-1}{2} \tag{3.15a}$$

$$\bar{\phi}(k) = \phi(k) - \bar{\theta}_r - \bar{\theta}_s k \tag{3.15b}$$

where

$$\bar{f} = m(1) \tag{3.15c}$$

$$\bar{\theta}_r = \frac{\phi(1) + \phi(-1)}{2} \tag{3.15d}$$

$$\bar{\theta}_s = \frac{\phi(1) - \phi(-1)}{2} \tag{3.15e}$$

The rationale behind this normalization is that most of the energy of the contour is concentrated in the first harmonic at $k = \pm 1$, especially at $k = 1$. Here, $m(k=1)/N$ is the radius of the first-order circle traversing counter-clockwise and $m(k=-1)/N$ is the radius of the first-order circle traversing clockwise. When combined, they trace out an ellipse whose major chord length is $2\{m(1) + m(-1)\}/N$ and whose minor chord length is $2\{m(1) - m(-1)\}/N$. We have assumed here $m(1) > m(-1)$ so that the traverse direction of the contour is counter-clockwise. If $m(1) < m(-1)$, the contour traverses clockwise and must be reversed in direction by setting $m(k) = m(-k)$. ($m(-1)$ can be zero or very small, in which case higher-order terms must be examined for consistent normalization. Wallace gives a more refined discussion of this issue in [55].) The major axis inclination of this ellipse is θ_r from the x-axis and θ_s is the polar angle to the starting point (n=0) from the major axis (see Figure 3.2). This first-order ellipse can be described by

$$c_1(n) = \frac{1}{N}\{m(1)\, e^{j\left(\phi(1) + \frac{2\pi}{N}n\right)} + m(-1)\, e^{j\left(\phi(-1) - \frac{2\pi}{N}n\right)}\}$$

$$0 \le n \le N-1 \tag{4.16}$$

Notice that $m(1)/N$ can be thought of as the *average radius* of the contour. All other circles for $k \neq 1$ simply deform the first-order circle in and out radially but never change the average radius. Thus Eq. (3.15a) normalizes the contour to *unit average radius*. (The real radius of the first-order circle is m(1)/N. However, using $\bar{f} = m(1)$ instead of $\bar{f} =$

$m(1)/N$ actually factors out N, making the matching independent of the transform size). It can be seen from Figure 3.2 that Eq. (3.15b) orients this first-order ellipse such that its major axis coincides with the x-axis. Also the contour starts on the x-axis when n=0 since $\bar{\phi}(1) = \bar{\phi}(-1) = 0$. All the higher-order circles (or ellipses) are reoriented relative to this first-order ellipse in such a manner that the shape of the contour is preserved.

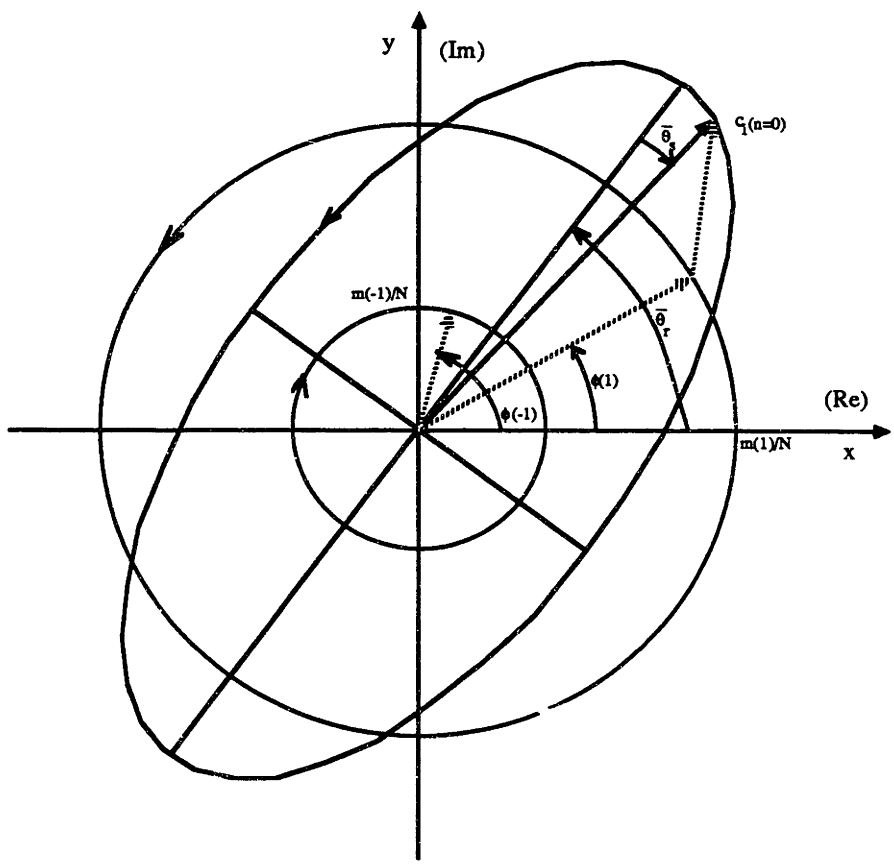


Figure 3.2
First-order ellipse $c_1(n)$

One problem with this normalization scheme is that there is a 180-degree ambiguity in rotation. This arises because an ellipse is two-fold symmetric under rotation. More work is needed to determine how to incorporate the higher-order terms (k>1) to resolve this ambiguity. However, because the energy in the higher frequency coefficients decreases rapidly, incorporating higher-order terms does not always resolve the ambiguity. Instead, we take a simple-minded

approach where the two-fold ambiguity is resolved by making two MSE measurements and then picking the orientation that corresponds to the smaller of the two. This has proven to work reliably.

3.3 Contour Interpolation

It is easier to divide the problem of interpolation into two and discuss each one separately: spatial interpolation within a frame and temporal interpolation between frames. Spatial interpolation increases the sampling density of the contour. Since decimation, which decreases the density, is closely related, we will discuss both spatial interpolation and decimation. Ordinary temporal interpolation of continuous-tone images guided by motion information can be troublesome if the magnification and rotation information is included in addition to the translation. However, in the case of processing contours, it becomes relatively easy if we use the circular-transform properties.

3.3.1 Spatial Interpolation/Decimation of Contours

Interpolation and decimation of 1-D signals can be easily carried out, either in the signal space or in the frequency domain. Suppose we have an N-point contour and would like to interpolate/decimate to get an L-point contour. When $L>N$ (interpolation), we will not lose any information, but when $L<N$ (decimation), we will need to band-limit the signal in order to avoid aliasing. In the time-domain (signal-space) approach, an appropriate interpolation/decimation filter is convolved with the signal, and the resulting signal is suitably resampled to L points. In the frequency-domain approach, the computation is simpler since the convolution is replaced by windowing. The spectrum is first multiplied by a periodic window function (transform of the interpolation/decimation filter) and the resulting frequency replicas are displaced relative to each other to have a period of L (see Figure 3.3).

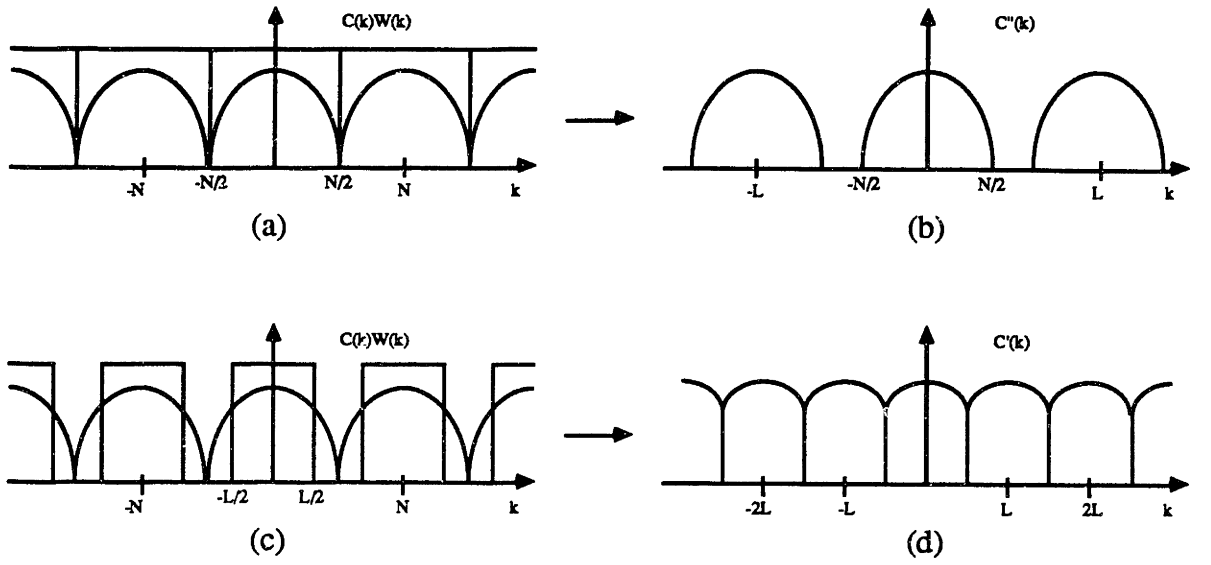


Figure 3.3
(a) and (b) : Interpolation process in frequency
(c) and (d) : Decimation process in frequency

Here we have chosen to use the rectangular window which corresponds to the ideal low-pass filter. Because the closed contour signal is truly periodic, the ILPF is both feasible to implement and desirable; it produces no undesirable ringing and has the maximum bandwidth possible. Besides, almost no real computation is required to implement this scheme (multiplications by 0 or 1). The sequence of plots in Figure 3.4 shows an example where a contour is gradually decimated and interpolated back in the circular-transform domain. Since the interpolation was performed from the decimated contour, the interpolated contours are somewhat band-limited but still retain the general shape. In fact, it can be said that the contour band-limited in this manner is usually the best approximation (in the MSE sense as given in Eq. (3.12)) to the original contour, since the magnitude of the Fourier coefficients usually decreases rapidly at a rate proportional to $1/k^2$ [54].

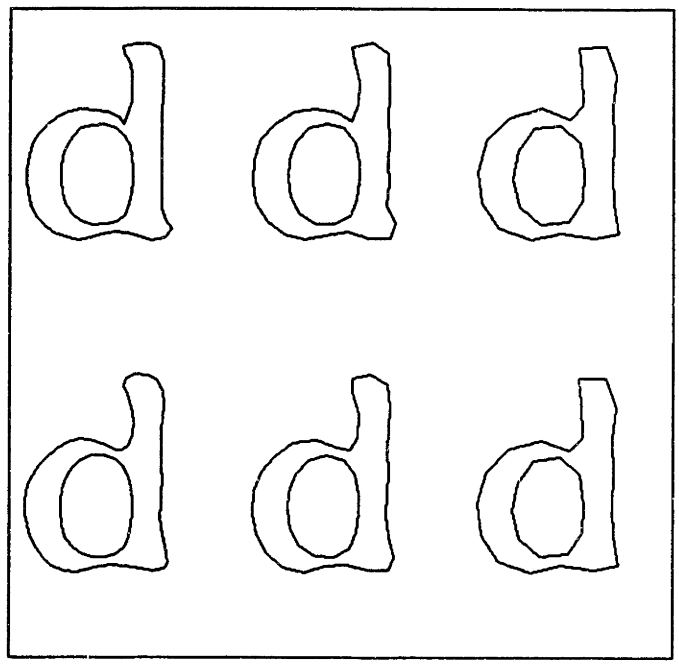


Figure 3.4
Decimated and interpolated contours processed
in the transform domain.
(a) original, L=52, (b) decimated, L=30, (c) decimated, L=20
(d) same as (c), L=20, (e) interpolated from (d), L=30,
(f) interpolated from (d), L=52

3.3.2 Temporal Interpolation of Contours

Once a match is established between two contours from two successive key frames, we would like to compute the in-between frames to increase the frame rate. Given the motion parameters, one can geometrically transform the contour in frame 1 such that it gradually changes from frame 1 to frame 2. However, if the match is not perfect or the motion parameters contain error, by the time it gets to frame 2 the transformed contour will not exactly coincide with the contour in frame 2. The situation is the same if the contour in frame 2 is used instead to move backward in time. The imperfect match and motion estimation error can be caused by noise in the picture

sequence, spatial and intensity quantization effects, deformation of the contour over time, etc.

In order to interpolate the contour smoothly from frame 1 to frame 2 even in the presence of noise and deformation, we propose to interpolate the circular transform coefficients. A bilinear interpolation is assumed here but higher-order interpolation using more than 2 frames is also possible. Recall that when two contours are to be matched, the transform coefficients are normalized so that they are as close as possible at each frequency. If the contours match exactly, the normalized coefficients are identical at each frequency, so we only need to add back the appropriate translation, magnification and rotation values to go from contour 1 to contour 2, or anywhere in between. If the two contours are not identical but have a similar shape, the normalized transform coefficients will be similar so that we can linearly interpolate between each pair of corresponding coefficients and still retain the general shape.

Let $\bar{C}_1(k)$ and $\bar{C}_2(k)$ be the normalized transform of the two matching contours in frame 1 and frame 2. Suppose frame 1 is at time t=0 and frame 2 is at time $t=T$ where T is the frame period. Then the linearly interpolated contour at time $t=\tau$ for $0 \le \tau \le T$ has the normalized transform $\bar{C}_\tau(k)$.

$$\bar{C}_\tau(k) = (1-\tau)\bar{C}_1(k) + \tau\bar{C}_2(k) \qquad 1 \le |k| \le \frac{N-1}{2} \tag{3.17}$$

where

$$\bar{C}_i(k) = \bar{m}_i(k)e^{j\phi_i(k)}$$

If $\bar{f}_1$, $\bar{\theta}_{r1}$ and $\bar{\theta}_{s1}$ were used to normalize the contour 1 and $\bar{f}_2$, $\bar{\theta}_{r2}$ and $\bar{\theta}_{s2}$ were used to normalize the contour 2 as in Eqs. (3.15), we linearly interpolate these values also:

$$\bar{f}_\tau = (1-\tau)\bar{f}_1 + \tau\bar{f}_2 \tag{3.18a}$$

$$\bar{\theta}_{r\tau} = (1-\tau)\bar{\theta}_{r1} + \tau\bar{\theta}_{r2} \tag{3.18b}$$

$$\bar{\theta}_{s\tau} = (1-\tau)\bar{\theta}_{s1} + \tau\bar{\theta}_{s2} \tag{3.18c}$$

Because the linear-phase term does not affect the shape or the orientation of the contour we need not use $\theta_{s\tau}$. The interpolated contour is then

$$C_\tau(k) = \bar{f}_\tau e^{j\bar{\theta}_{r\tau}} \bar{C}_\tau(k) \qquad 1 \le |k| \le \frac{N-1}{2} \tag{3.19a}$$

where $\bar{C}_\tau(k)$ is given in Eq. (3.17). Of course the translational component is handled separately so that

$$C_\tau(k) = (1-\tau)C_1(k) + \tau C_2(k) \qquad \text{for } k=0 \tag{3.19b}$$

One minor detail we have not included so far concerns the spatial interpolation/decimation necessary when the transform sizes of the two contours are not the same. Suppose the contour 1 is N_1 points long and the contour 2 is N_2 points long. If we desire the interpolated contour to be N_τ points long, we simply compute only N_τ points of $C_\tau(k)$ for $1 \le |k| \le (N_\tau - 1)/2$. Any coefficients missing in the computation (i.e. for $|k| > (N_\tau - 1)/2$ are set to zero. This is consistent with the interpolation/decimation process described above.

In deciding the value of N_τ it seems to be safe to pick a value between N_1 and N_2. Therefore we also interpolate the transform length. The complete formula for temporal interpolation of contour is given below.

$$C_\tau(k) = (1-\tau)\frac{N_\tau}{N_1} C_1(k) + \tau \frac{N_\tau}{N_2} C_2(k) \qquad \text{for } k=0 \tag{3.20a}$$

$$C_\tau(k) = \bar{f}_\tau e^{j\bar{\theta}_{r\tau}} \bar{C}_\tau(k) \qquad \text{for } 1 \le |k| \le \frac{N_\tau - 1}{2} \tag{3.20b}$$

where

$$\bar{C}_\tau(k) = (1-\tau)\bar{C}_1(k) + \tau\bar{C}_2(k) \qquad \text{for } 1 \le |k| \le \frac{N_\tau - 1}{2} \tag{3.20c}$$

$$\bar{f}_\tau = (1-\tau)\frac{N_\tau}{N_1}\bar{f}_1 + \tau\frac{N_\tau}{N_2}\bar{f}_2 \tag{3.20d}$$

$$\bar{\theta}_{r\tau} = (1-\tau)\bar{\theta}_{r1} + \tau\bar{\theta}_{r2} \tag{3.20e}$$

$$N_\tau = (1-\tau)N_1 + \tau N_2 \tag{3.20f}$$

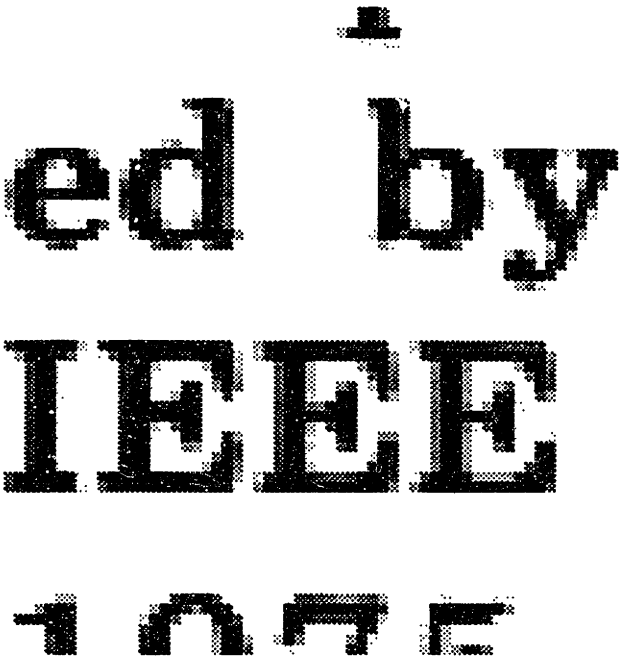
(The factors N_τ/N_1 and N_τ/N_2 appearing in Eq. (3.20a) properly account for the different transform sizes involved).

Again, this temporal interpolation process takes relatively little computation if we make use of the intermediate results in the matching process (e.g. normalized transform values, parameters used for the normalization, etc).

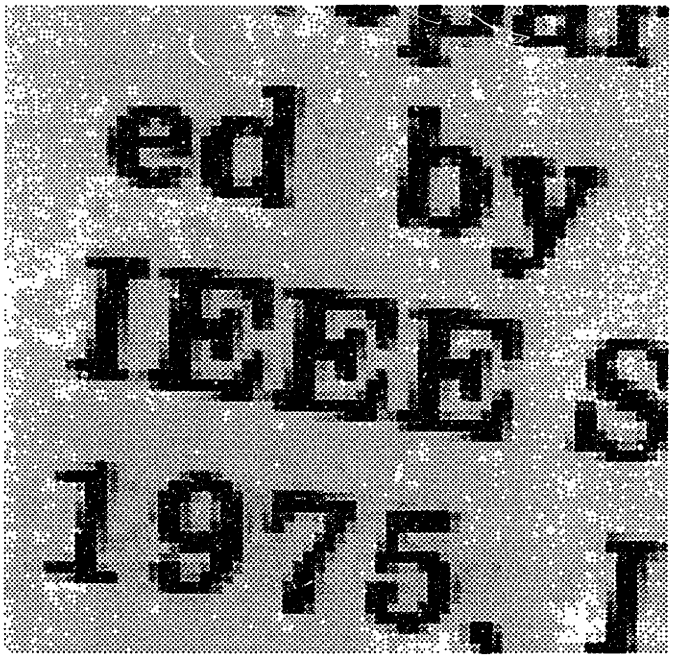
3.4 Experiment

Experimental programs were set up to extract contours from a pair of image frames and perform the matching and interpolation based on the previous discussions. Figures 3.5(a) and 3.5(b) show two images from the IEEE test chart scanned in at two different orientations and magnifications. Figures 3.6(a) and 3.6(b) show the contours extracted from these two images by an iso-luminance contouring process. Due to different density settings used at scanning and due to a moderate subsampling (by a factor of 4), the corresponding letters in the images are not identical. Therefore they represent a sequence of two successive frames undergoing translation, magnification, rotation, and small distortion in shape. Figure 3.7 shows the interpolated frame halfway between Figures 3.6(a) and 3.6(b). The contours that did not match sufficiently well in the matching process were excluded from the interpolation.

The current matching program implements a crude scoring system that involves four parameters: MSE, displacement, magnification factor, and rotation angle. Basically, it looks for a contour that requires the minimum amount of transformation and has a small MSE at the same time. (Actually, $\log(mse_{ij} + 1)$ was used instead of mse_{ij} because the MSE for a correct match was 2 or 3 orders of magnitude lower than the MSE for an incorrect match). It declares a no-match if any of the four parameters exceeds a certain threshold (for example, the contours occluded at the image boundary.)

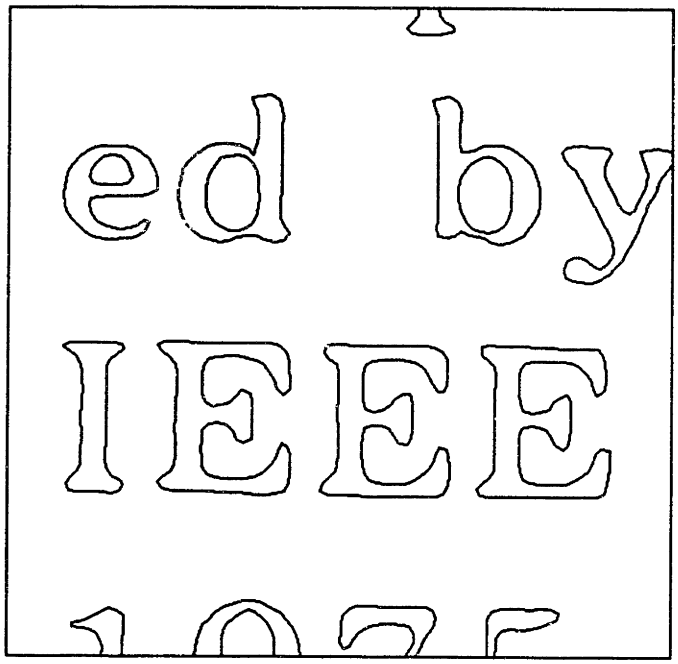


(a)

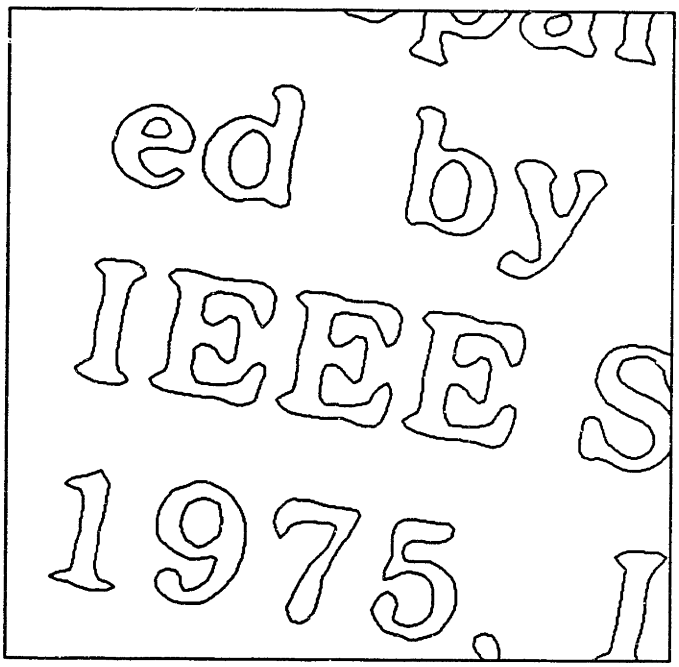


(b)

Figure 3.5
Two images used for matching and interpolation



(a)



(b)

Figure 3.6
Contours extracted from Figures 3.5(a) and 3.5(b)

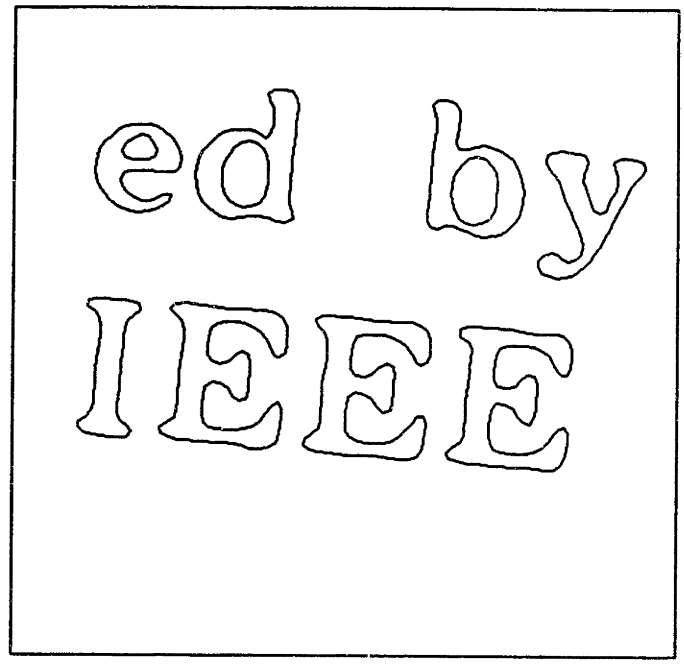


Figure 3.7
Interpolated from Figures 3.6(a) and 3.6(b)

CHAPTER 4. OPEN CONTOURS

A good contour-matching algorithm must provide an accurate estimate of the translation, rotation, and magnification parameters, given the description of two contours. Furthermore, it must provide the location of the overlap and a measure of the degree of matching -- how similar the overlapping segments are. Ideally it should be robust enough to tolerate a small amount of noise in the form of minor deformation as well as small quantization/sampling error.

The transform method that was developed in Chapter 3 works very well for closed contours, even in noisy conditions, because it extracts the global features of the contour. The low-frequency coefficients define the gross shape of the contour while the higher-frequency coefficients describe the details. The transform method is inherently robust because the coefficients are computed by averaging over many data points (i.e. all samples along the contour). However, the shortcoming of such a global feature extractor is that the feature *parameters* cannot accurately describe local changes along the contour. In other words, any local change of the contour shape can affect every feature parameter. For example, a sharp spike-like bump on an otherwise smooth contour can have a similar effect as an impulse function which has energy at all frequencies, thereby perturbing all feature parameters. This global effect of a local change makes the transform method unsuitable for partial-contour matching because the end points of a partial (open) contour act like sharp steps when the contour is forced closed. Even if the contour can be closed without introducing sharp corners, the section of the contour that was added to close the open contour adds significant energy to the transform and make the resulting parameters unreliable. Perhaps the most devastating problem with applying a transform method to partial contours is that the periodicity of the contour is lost when the length of the missing section of the contour is unknown. If the length of the added section is longer, the feature parameters shift toward the high frequency. If it is shorter, the parameters shift toward the low frequency. Basically, the frequency axis gets scaled as the relative

length of the contour changes. In this chapter, we present techniques to match open contours using local feature parameters.

4.1 Previous Work

Despite these fundamental difficulties there has been some effort to extend the transform method to handle partial contours [71 & 73]. On the other hand, Gorman et. al. [72] proposed to combine the transform method with a dynamic programming technique. They use a very simple segmentation method to divide the contour into a number of segments. To obtain a partial rotational invariance, they find two points along the contour that are farthest apart from each other. The contour is initially split into two at these points. Further segmentation is performed at the point farthest from the line connecting the two end points of the segment. The procedure is repeated recursively until the distance of the farthest point falls below a set threshold. The resulting segments are transformed into Fourier descriptors. The transform coefficients are then compared to form an intersegment distance table as if taking an outer product of two vectors. Basically, every contour segment of the first contour is compared against every segment of the second contour, forming a matrix of similarity measures. A dynamic programming technique is then employed to search for the best diagonal path through the matrix that has the minimum overall distance measure. Such a path provides a correspondence between matching segments. This procedure is repeated for all known shapes, or reference templates, and the template that produces the minimum overall distance is selected as the recognized object. When the contour is not closed, it is arbitrarily closed by a straight line. Since each segment is essentially an open contour in itself, the transform was taken so that the segment is effectively closed by traversing it backward, thus making it twice as long. However this does not make the transform size twice as long since only the even coefficients need to be computed due to an even symmetry.

As expected, this technique is rather sensitive to the segmentation process and does not provide any accurate estimate of the rotation or magnification factors or the overlap locations. Increasing the number of segments, thereby reducing the segment

size, can improve these estimates. But, as the segment size gets smaller, the distance measure becomes less reliable and the problem gets compounded by a large search space for the dynamic programming algorithm. A large search space makes it more prone to make wrong decisions when searching through the intersegment distance table for the optimum path. Although this technique was devised for recognition purposes, it bears many structural resemblances to the proposed matching technique that will be described shortly.

4.2 Proposed Approach

In order to describe the proposed technique properly, let us start by developing the basic framework. Beginning with the same contour representation as in Chapter 3, we define the curvature representation. Then the idea of the distance matrix is presented along with a dynamic time-warping (DTW) technique. Instead of using DTW, we propose to use the Hough Transform to find the best straight line path through the distance matrix. It will be shown that the magnification factor is found as the slope of the straight line and the overlap location as well as the rotation and translation factors can eventually be derived.

4.2.1 Curvature Representation

As in the transform method in Chapter 3, let us parameterize the partial contour as a function of arc length such that we get a one-dimensional, complex-valued function of arc length s.

$$c(s) = x(s) + jy(s) \qquad \alpha \le s \le \beta \tag{4.1}$$

α and β are the lower and upper bounds of s, representing the end points of the open contour $c(s)$. Unlike in Chapter 3, $c(s)$ need not be assumed cyclic in s. The curvature signal is obtained by taking the derivative of the tangent function $T(s)$.

$$K(s) = \frac{dT(s)}{ds} \qquad \alpha \le s \le \beta \tag{4.2}$$

where

$$T(s) = \angle \frac{dc(s)}{ds} = \arctan\left[\frac{dy(s)}{ds} / \frac{dx(s)}{ds} \right] \tag{4.3}$$

Notice that the translational component is removed by taking the tangent $T(s)$ along the contour, and the rotational component is removed by taking the curvature $K(s)$. These two simple steps extracts the *shape* information of the contour. However, $K(s)$ still contains the magnification component that needs to be determined

when comparing against another contour. Even after the relative magnification factor is found, one of the contours has to be shifted in s until the two contours come into a proper alignment. This shift in s can be used to estimate the relative rotation factor. Then the translation can be determined from the aligned segments. It should be noted that, because of two differentiation steps to obtain $K(s)$, we must eventually address the noise issue. However, the algorithm is robust enough to tolerate some noise, such as due to sampling and quantization, as well as minor deformation of the contour.

4.2.2 Distance Matrix

Given two contour signals $c_1(s)$ and $c_2(s)$, define a distance function over a region R.

$$D(s_1,s_2) = P[\, F(c_1(s_1)), F(c_2(s_2))\,] \tag{4.4}$$

such that

$$D(s_1,s_2) = 0 \quad \text{when } F(c_1(s_1)) \neq F(c_2(s_2))$$

$$D(s_1,s_2) = 1 \quad \text{when } F(c_1(s_1)) \neq F(c_2(s_2))$$

The region of support R is a rectangular region over s_1 and s_2 such that $\alpha_1 \leq s_1 \leq \beta_1$ and $\alpha_2 \leq s_2 \leq \beta_2$. $D(s1,s2)$ is undefined outside R. α_1 and β_1 are the end points of $c_1(s)$ and α_2 and β_2 are the end points of $c_2(s)$. F is a *feature* function that is derived from the contour and measures the local characteristics of the contour. Later we will use the curvature $K(s)$ or the tangent $T(s)$ as the feature function. The function P is a very simple decision function that is zero if the local feature values are identical and non-zero if they are different.

The rationale for defining such a distance function is to find a warping function by using a technique such as the Dynamic Time Warping (DTW) on the two contours. If the contours $c_1(s)$ and $c_2(s)$ match and share a common matching segment, then there exists a warping function $W(s)$ such that

$$F(c_1(s)) = F(c_1(W(s))) \qquad \text{for } \rho \leq s \leq \gamma. \tag{4.5}$$

The warping function $W(s)$ warps $c_2(s)$ to match the shape of $c_1(s)$. It implicitly contains the magnification and the rotation information. For example, if the contours match perfectly without deformation (rigid transformation), $W(s)$ must be a straight line:

$$W(s) = as + b \tag{4.6}$$

The slope a of the line equals the magnification factor. The intersect b can be used to compute the rotation by finding ρ and γ, the end points of the matching contour segment. For a more general case where small deformation of the contour is allowed, $W(s)$ is not a straight line but a monotonically increasing function of s (monotonically decreasing if one of the contours is reversed in s). However, once a deformation is allowed, $W(s)$ cannot be uniquely defined since no unique mapping exists between the contours (unless we impose a restriction such as the smoothness constraint used by Hildreth).

If the contours $c_1(s)$ and $c_2(s)$ were analytic functions of s, it is possible to find suitable functions for F and P so that $D(s_1,s_2)$ becomes a two-dimensional analytic function. In that case, one can solve for $W(s)$ as a root locus of $D(s_1,s_2)$. In reality, we cannot require $c_1(s)$ and $c_2(s)$ to be analytic. $c_1(s)$ and $c_2(s)$ will have to be represented as sampled functions $c_1(n_1)$ and $c_2(n_2)$, and consequently $D(s_1,s_2)$ has to be represented as a two-dimensional sampled function:

$$D(n_1,n_2) = P(F(c_1(n_1),F(c_2(n_2)) \tag{4.7}$$

defined for $\alpha_1 \le n_1 \le \beta_1$ and $\alpha_2 \le n_2 \le \beta_2$.

We will call $D(n_1,n_2)$ the *distance matrix* for $c_1(n_1)$ and $c_2(n_2)$. Instead of solving for $W(s)$ analytically, we must look for a heuristic method similar to the DTW algorithm.

4.2.3 Dynamic Time Warping

Dynamic time warping is a dynamic programing technique mainly used for time-scale registration of speech [74]. In isolated word recognition, the test pattern and the reference pattern are not

perfectly aligned in time. The spoken word may be shorter or longer than the reference word. Also a proper alignment often requires a local compression or expansion requiring a nonlinear time warping. So the DTW is formulated as a path finding algorithm over a finite two-dimensional grid. Each grid point represents a distance between two *feature* vectors from each of the test and the reference patterns. The feature vectors are typically the local autocorrelation coefficients computed over a frame of fixed length. A dynamic programming technique is then used to find the optimum warping function $W(n)$ that minimizes a certain overall distance measure.

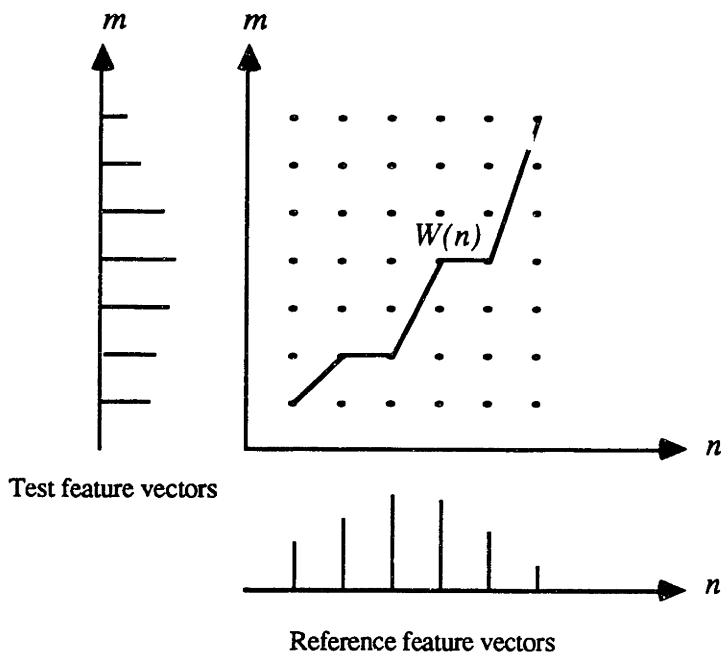


Figure 4.1
Dynamic Time Warping (DTW)

Because of the nature of speech generation, the feature vectors vary slowly enough to permit a reasonable frame length (10-50 ms) and the resulting feature vectors are relatively accurate because they are derived from many sample points. This also makes the location of the frame boundary less critical to the performance. When applied to the matching of contours, as done by Gorman et. al., it poses a significant challenge because a small shift in the segmentation (frame) boundary can produce radically different shapes.

(a) contour (b) segmentation 1 (c) segmentation 2

Figure 4.2
Contour segmentation

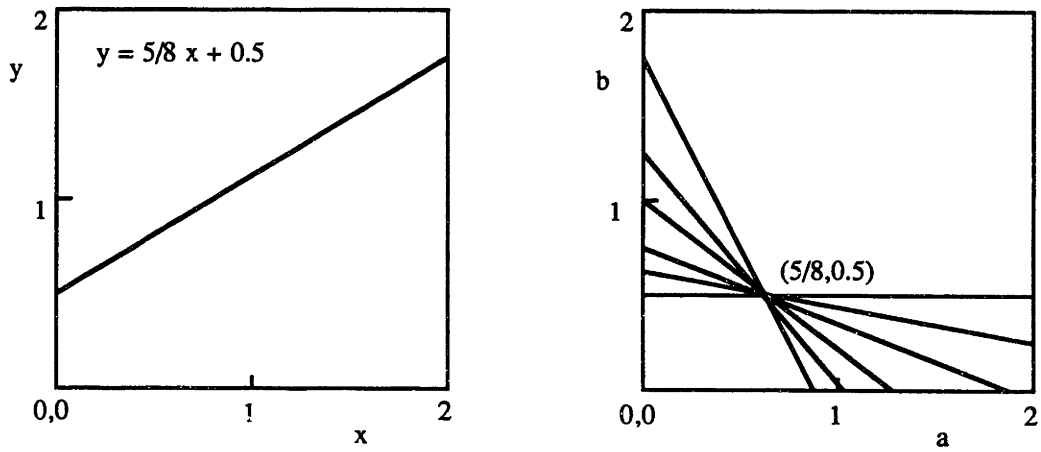
Segmenting a contour consistently is a very difficult problem. Also, as stated earlier, we need an algorithm that not only makes a similarity measure to see if the test pattern matches the reference, but also provides an accurate estimation of the end points of the matching segment and subsequently the rotation and magnification estimates. Obviously this is not possible if the segment size is large. On the other hand, reducing the segment size makes the feature vectors unreliable and increases the search space for the dynamic programming.

In order to avoid segmenting the contour and improve the end points estimation, we elect to use a point function such as the curvature function defined above as the feature vector. This does increase the size of the distance matrix over which the DTW has to work. Also the distance measure is very coarse since a curvature value on one contour may have many matching points on the other contour and vice versa. Although DTW is a very powerful method, it cannot be expected to work reliably when provided with a poor data set. With a few simplifying assumptions, the proposed technique uses the Hough Transform to find the warping function over the curvature distance matrix.

4.2.4 Hough transform

The Hough transform was originally devised to find a straight line in a noisy image [75]. It works by mapping a line in the image space onto a point in the parameter space. However a line in the parameter space in turn maps onto a point in the image space. The

line finding algorithm first identifies candidate points in the image that may belong to the line we are looking for, usually by using a thresholding process. Then for each such point a corresponding line is drawn in the parameter space. If all points belong to a straight line, then they would produce lines in the parameter space with a common intersect. This intersection point maps to the line in the image space.



(a) A line in image space

(b) Hough parameter space

Figure 4.3
Hough Transform

Suppose we are looking for a straight line of the form

$$y = ax + b \tag{4.8}$$

The parameters we must find are the slope a and the intersect b. A line in the parameter space is then represented by

$$b = -xa + y \tag{4.9}$$

Given a (x,y) pair in the image, equation (4.9) maps out a line in the Hough parameter space. Every (x,y) pair in the line must satisfy the equation (4.9). Therefore they map out lines with a common intersect at (a,b). In order to find this intersection point, a two-dimensional accumulator array is set up and the value of the cell the line passes by is incremented. A cell with the highest accumulated value is selected and its location gives the desired a and b values.

The Hough transform is very robust even when the input data set is very noisy. Erroneous points outside the line produce lines with no common intersect and cannot form a false peak in the accumulator array. The points on the line need not be connected in the image (e.g. dotted lines or disjoint lines) since the algorithm works on a point-by-point basis.

4.3 Curvature distance

Intuitively, the curvature of a contour captures all of the shape characteristics of the contour; a straight line segment has a zero curvature, a sharp corner has an impulse-like curvature, a smooth circular segment has a constant curvature, etc. Looked at another way, a contour can be completely reconstructed from its curvature if the magnification, rotation and translation factors are provided. Being a one-dimensional function, the curvature can be treated like a time-domain signal. In fact, the curvature can be smoothed by a linear filter and the resulting contour becomes smooth. If we make an analogy to a speech signal, a good way to match two curvature signals would be to perform a time-scale registration as in the isolated word recognition problem. If, after the registration, the two curvatures still differ considerably, they are rejected. If they match closely, they are declared a match and the parameters used in the registration are used to compute the relative magnification (scale) and rotation (time shift) factors. In order to perform the time-scale registration we utilize the concepts developed above: the curvature distance matrix and the Hough transform.

Referring to the Eqs. (4.2) and (4.3) we redefine the curvature and tangent functions for the discrete case:

$$K(n) = T'(n) - T'(n-1) \tag{4.10}$$

where

$$T'(n) = \arctan\left[\frac{y(n+1) - y(n)}{x(n+1) - x(n)} \right] \tag{4.11}$$

and

$$T(n) = \frac{T'(n) + T'(n-1)}{2} \tag{4.12}$$

The reason for defining two different tangent functions $T(n)$ and $T'(n)$ is to assign the tangent angle *at* the vertice $c(n)$ to avoid a half-sample

spacing offset in the calculations. $T'(n)$ is the tangent angle of the line connecting the contour points $c(n)$ and $c(n+1)$. $T(n)$ is the average of the angles of the two adjacent line segments since the tangent angle is not defined at the vertice $c(n)$. Although curve fitting can be employed to improve the tangent and curvature estimates, the improvements do not have a significant effect on the matching because of the sampling effect. From Eq. (4.7) we define the curvature distance matrix as follows:

$$D(n_1,n_2) = P[\, K_1(n_1),\, K_2(n_2)\,] \tag{4.13}$$

In view of the sampled nature of the curvature functions, we cannot enforce the same decision function P as in Eq. (4.4). Therefore the distance measure is somewhat relaxed by providing a continuous range of values rather than 0 or 1. A threshold value other than zero can be used to select candidate matching points. We use the absolute difference of the curvatures.

$$D(n_1,n_2) = P[\, K_1(n_1),\, K_2(n_2)\,] = |\, K_1(n_1) - K_2(n_2)\,| \tag{4.14}$$

The reason for selecting this distance measure, besides its computational simplicity, is that almost any reasonable distance function performs similarly due to a rather large noise in the curvature measure. As a comparison, a logarithmic difference of the form

$$D(n_1,n_2) = |\, \log(K_1(n_1)) - \log(K_2(n_2))\,| \tag{4.15}$$

has been tried. The reasoning was that, since the contour is uniformly sampled in arc length, the high curvature segments are effectively undersampled and the low curvature segments are oversampled. Therefore the curvature measure is more accurate for the low values than for the high values. Taking the logarithmic difference allows higher error in high curvature values by providing a reasonably high log threshold (of course we must insure that zero curvature values do not map to the negative infinity by providing another threshold). In practice, however, the curvature difference for low curvature pairs (even when they are supposed to match) often exceeds the curvature

of smaller of the two, requiring an additional threshold value (in linear scale) to include these small curvature pairs into the Hough transform. Because small curvature values are more common than large values, this second threshold dominated over the log threshold. This second threshold has a similar roll as the threshold for the linear curvature difference, and consequently it did not perform any better than the case using the linear difference.

The reason for a poor curvature estimate for the low values is that we are not taking advantage of that information. If it is known a priori that the local curvature value to be estimated is small, we could use many neighboring data points to increase the accuracy (e.g. using a high-order curve fitting). For high-curvature points, we have to use fewer data points to preserve the locality. Rather than designing an elaborate adaptive scheme to improve the curvature measure, we choose to use the simple distance function in Eq. (4.14). We designate this threshold λ. No extra effort was spent to find the optimum threshold because the range of λ that worked well was quite broad and it was difficult to come up with a good analytic criterion for selecting it.

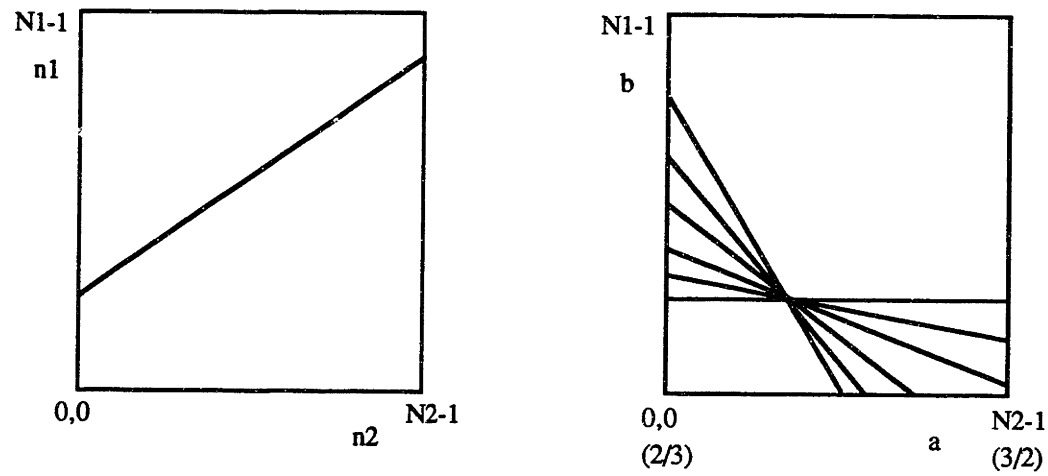
Until now the effect of magnification on the curvature has not been addressed. If a contour is magnified, it becomes longer and the curvature is scaled down linearly by the magnification factor m ($m>1$ if $c_1(n)$ is larger than $c_2(n)$ and vice versa). Ideally the distance measure has to be modified to incorporate this effect.

$$D(n_1,n_2) = | K_1(n_1) - K_2(n_2)/m | \tag{4.16}$$

Initially m is not known, so it is set to 1. After taking the Hough transform, the magnification factor m can be estimated. This m can be used in the next iteration to improve the distance measure. However, experiment has shown that in most cases the first estimate of m is accurate enough to not require a second iteration if the distance threshold λ is set high enough and the true value of m is between 2/3 and 3/2. This range permits 50% variation in magnification, which is adequate in most practical cases.

4.3.1 Hough Accumulator Array

As stated earlier, the Hough parameter space has to be sampled and replaced by an accumulator array in order to implement a two-dimensional histogram. The extent of the accumulator array can be limited by specifying ranges for a and b values. Since a is the estimate for m, we let $2/3 \leq a \leq 3/2$. b specifies the overlap location of the matching segment so it need not be larger than the extent of the contour. We let $0 \leq b < \max(N_1,N_2)$ where N_1 and N_2 are the lengths of the contours $c_1(n)$ and $c_2(n)$, respectively. Without loosing generality, we let $N_1 \geq N_2$. Then the dimensions of the distance matrix $D(n_1,n_2)$ is N_1 by N_2 and we choose the dimensions of the accumulator array to be the same, N_1 by N_2. That is, there are N_2 entries along the a axis and N_1 entries along the b axis. This way the spacing in b is the same as the sample spacing for the contours. Also the accuracy for a improves as the contours get longer so that the accuracy of b remains unchanged (which is important in finding the end points ρ anf γ).



(a) Distance matrix (b) Hough accumulator array

Figure 4.4
Distance matrix and Hough accumulator array

For each candidate point in the distance matrix, the Hough transform is performed using a simple line-drawing algorithm in the accumulator array. The value of each cell the line passes over is incremented to update the histogram. After all points are

transformed, the peak of the histogram is searched to find a and b. This value of a becomes the estimate for m. From a and b, two end points of the overlap segment can be found by locating the crossing points of the line with the boundaries of the distance matrix. For case (a) in Figure 4.5, $c_1(n)$ is a sub-segment of $c_2(n)$ so that the overlap segment is identical to $c_1(n)$. For case (b), one end point is contained in $c_2(n)$ and the other contained in $c_1(n)$. Case (c) is similar to case (b), but poses a problem since the line is mapped outside the accumulator array (because b is negative).

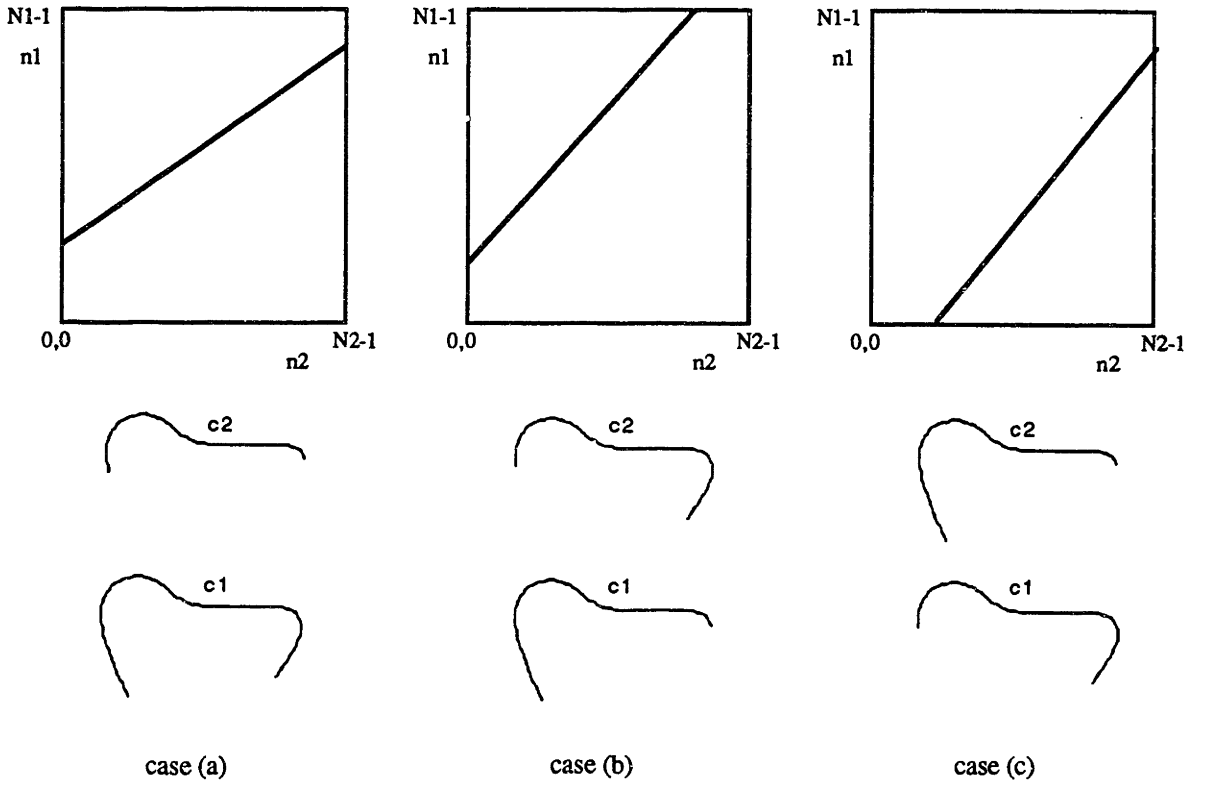
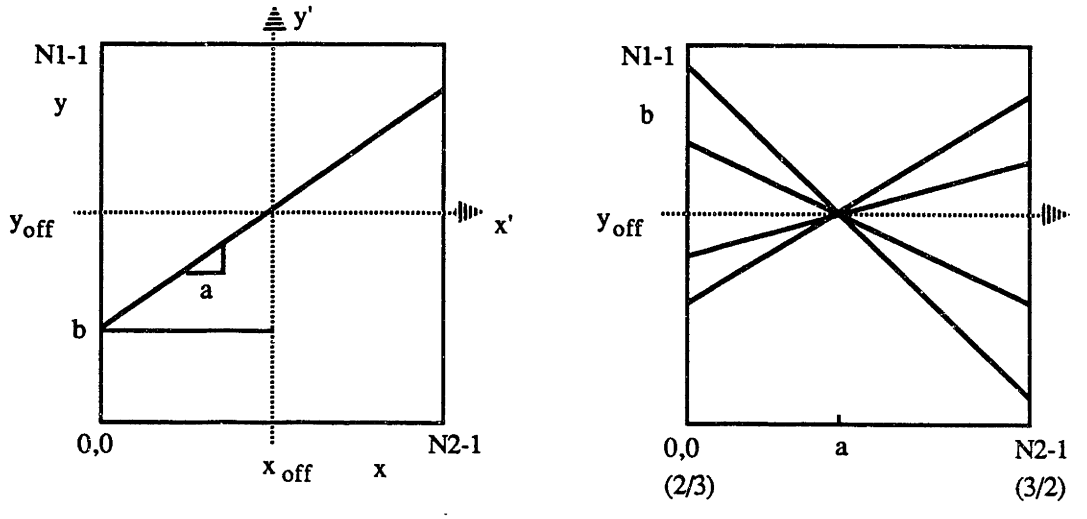


Figure 4.5
3 cases of overlap for open contours

Two remedies exist for this problem. Unfortunately both methods double the size of the Hough parameter space. One is to extend the Hough parameter space in b to $-N_1 \leq b \leq (N_1-1)$. The other is to swap the roll of $c_1(n)$ and $c_2(n)$ by transposing the distance matrix and computing a second Hough transform. The disadvantage of the former is that there is still a slim chance that b might be smaller than $-N_1$ when the overlap is very small and a is large. The disadvantage of

the later is that two histograms must be computed and the one with a lower histogram peak value is discarded. If the second histogram peak is higher, the estimate for m becomes $1/a$. For completeness, one would adopt this later approach to insure that all cases are covered, at the expense of more computation. Instead, we take an engineering solution by modifying the problem sightly to accommodate all three cases (a,b and c) with some restrictions.



(a) Distance matrix (b) Hough accumulator array

Figure 4.6
Modified distance matrix and Hough accumulator array

Instead of defining the origin of the distance matrix at the lower left-hand corner, we provide an offset in x by $x_{off}=1/2(N_2\text{-}1)$. The offset in y (y_{off}) is implicitly defined where the line crosses the y' axis (at x_{off}).

$$x_{off} = \frac{N_1 - 1}{2}$$

$$x' = x - x_{off}$$

$$y' = y$$

The axes in the Hough space are unchanged. If a is still restricted to between 2/3 and 3/2, from Fig. 4.6 we see that the real intersect b can be

derived from the slope a and the implicit offset y_{off}, both found from the Hough transform.

$$b = y_{off} - ax_{off} \tag{4.17}$$

The range of values for b can be found from the minimum and maximum values allowed for a using Eq. (4.17).

$$b_{ll} = 0 - \frac{2}{3}\frac{1}{2}(N_2-1) = -\frac{1}{3}(N_2-1) \tag{4.18a}$$

$$b_{ul} = (N_1-1) - \frac{2}{3}\frac{1}{2}(N_2-1) = (N_1-1) - \frac{1}{3}(N_2-1) \tag{4.18b}$$

$$b_{lr} = 0 - \frac{3}{2}\frac{1}{2}(N_2-1) = -\frac{3}{4}(N_2-1) \tag{4.18c}$$

$$b_{ur} = (N_1-1) - \frac{3}{2}\frac{1}{2}(N_2-1) = (N_1-1) - \frac{3}{4}(N_2-1) \tag{4.18d}$$

b_{ll}, b_{ul}, b_{lr}, and b_{ur} represent the b intersects corresponding to the lower-left, upper-left, lower-right, and upper-right corners of the Hough transform space (mapped back to lines in xy space).

From Figure 4.7 it is apparent that a line is not allowed to pass the shaded boundary. This means the overlap must be at least 50 percent of the length of the *shorter* contour, in addition to the restriction that the magnification must be less than 50 percent. These limitations are not very severe in practical situations and are imposed as a matter of convenience and not due to the limitations of the algorithm. The advantage gained here is that now the lines drawn in the Hough space can have positive slopes as well as negative slopes because x' is negative for half the entries in the distance matrix. This generally gives a better distribution (symmetric in a and b) of the histogram in the accumulator array to make the peak detection easier.

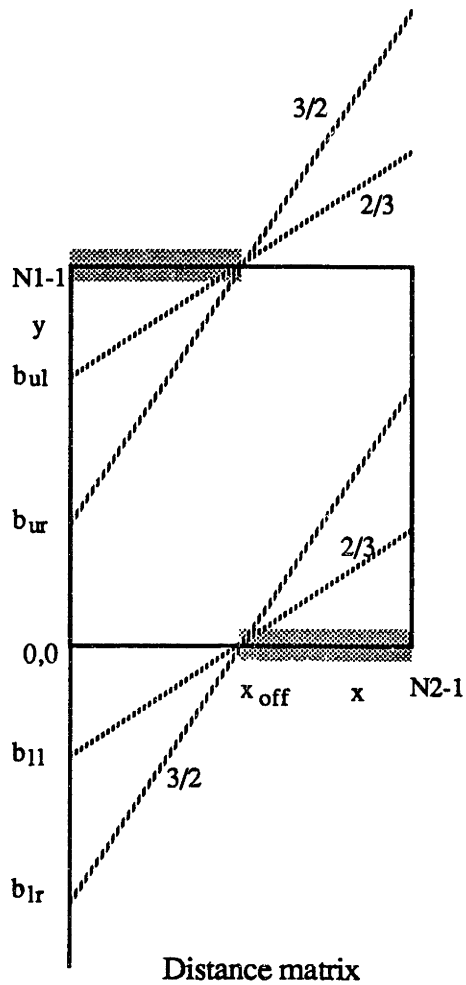
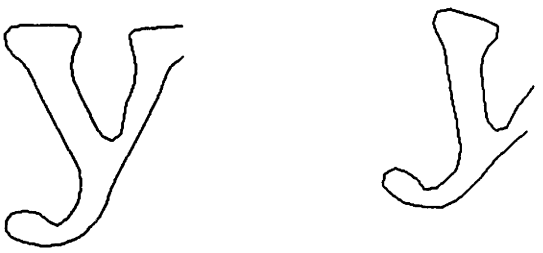


Figure 4.7
Allowed ranges for the warping function

4.3.2 Curvature Matching Example

In the following we show a curvature matching example for two open contours. Although this example represents a near ideal case in order to illustrate the principle, the contours do exhibit a small amount of dissimilarity due to the different filtering and sampling processes involved. The contour of a character y was extracted from a gray-scale image using the iso-luminance contouring process. Then it was successively translated, rotated, magnified (reduced in size), and finally clipped at the end points to derive the second contour. This second contour would correspond to an object undergoing a rigid transformation and gets occluded at the image boundary.



(a) (b)

Figure 4.8
Two contours used in the curvature matching example
(a) $c_1(n_1)$: N_1=64 (b) $c_2(n_2)$: N_2=55, m=1.111, θ_r=-0.1π

Figure 4.9 shows the curvatures for these two contours, among other things. The upper left quadrant shows two contours, properly resized to fit in the window. The lower left quadrant shows the x and y (real and imaginary) components of the contours. The upper right quadrant shows the contour transform magnitudes (normalized to fit in the window). Finally the lower right quadrant shows the curvatures. Notice that the two curvatures can be visually registered on the time scale, even though a perfect match is not possible.

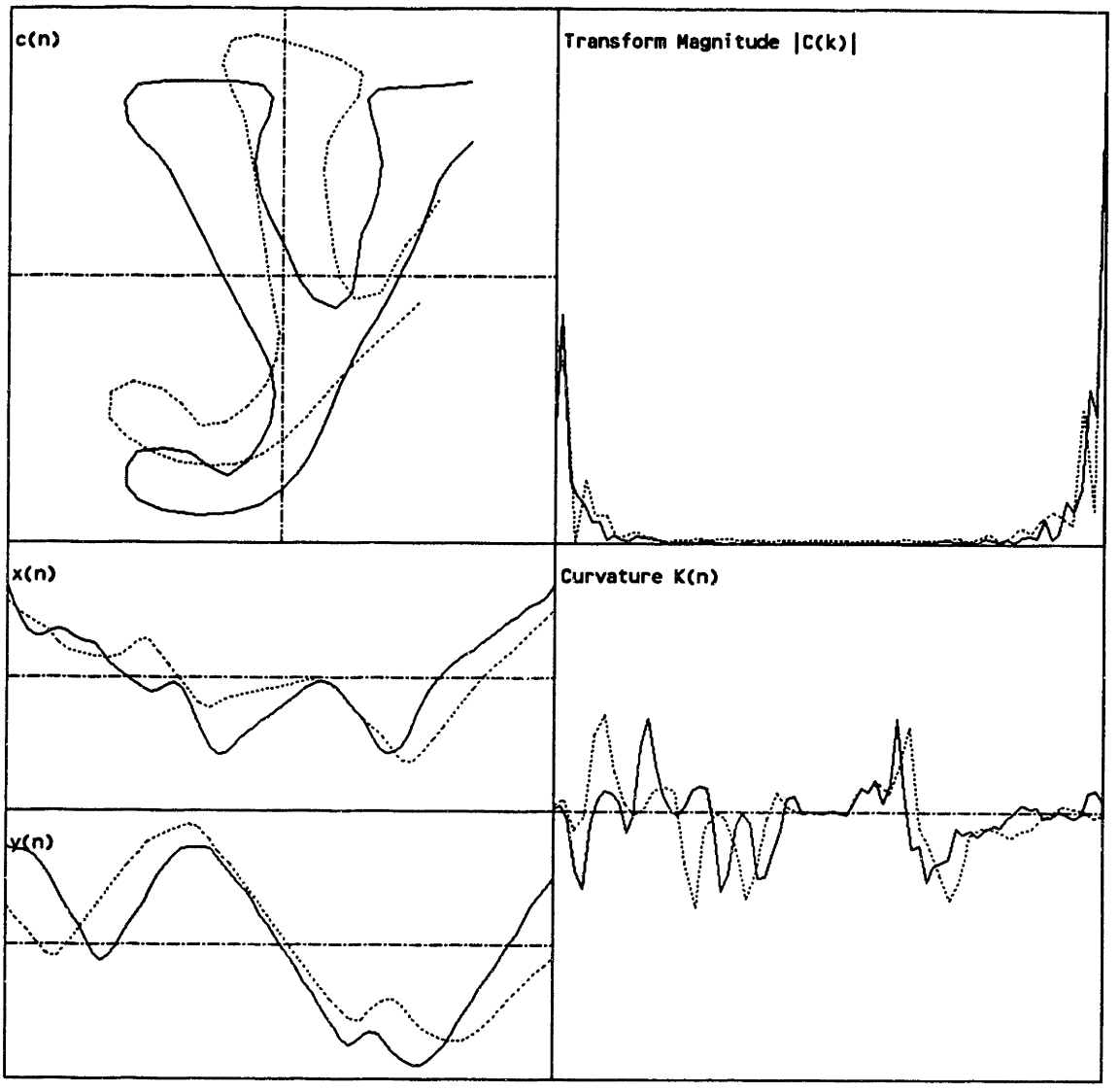
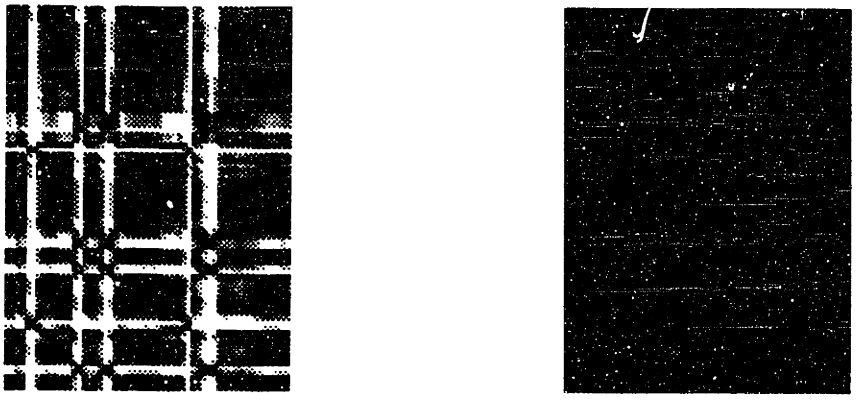


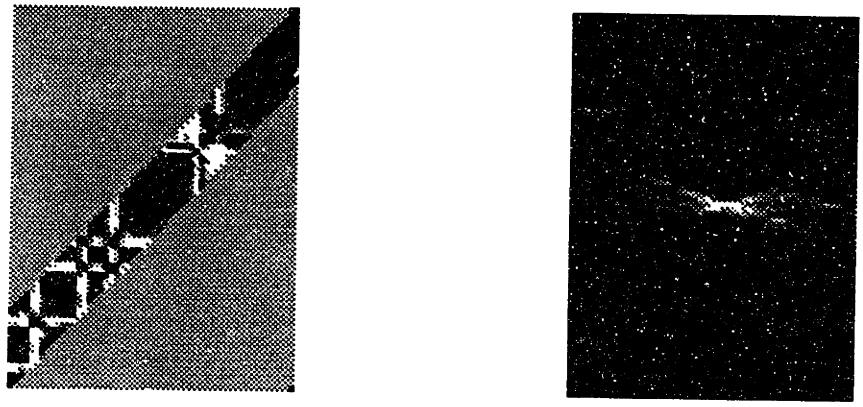
Figure 4.9
Curvature plots (lower right quadrant) for
c_1 (solid line) and c_2 (dotted line)



(a) (b)

Figure 4.10
Curvature distance matrix (a) and its Hough transform (b)

Figure 4.10(a) is the curvature distance matrix for $K_1(n)$ and $K_2(n)$ derived from $c_1(n)$ and $c_2(n)$ according to Eq. (4.10). Dark areas represent a small curvature distance and light areas represent a large distance (from 0 to π). It is apparent that the curvature at each point on the contour is not unique over the contour. This is expected, since a curvature is a scalar function derived as an instantaneous slope of the tangent angle. Each curvature value along the first contour has many matching points in the second contour. In general, a high-curvature point matches fewer points in the second contour than do the low-curvature points, simply because low curvature values are more common. It may seem hopeless to find a line through this distance matrix, but the corresponding Hough transform in Figure 4.10(b) does exhibit a definite peak. For this transform we have used $\lambda=0.01\pi$. Figure 4.12(a) gives a perspective plot of Figure 4.10(b) which also shows a prominent peak.



(a) (b)

Figure 4.11
Modified curvature distance matrix (a) and its Hough transform (b)

In order to show the uncluttered Hough transform, portions of the curvature matrix in Figure 4.10(a) were blocked off to isolate the area that does contribute to the peak in the Hough space (this area was determined from the initial transform in Figure 4.10(b)). This is shown in Figure 4.11(a). Figure 4.11(b) shows the resulting Hough transform, which has much of the clutter removed. However, this does not improve the detection of the peak since the points in the blocked-off area (medium shade in Figure 4.11(a)) map to lines that do not cross the peak point. Figure 4.12(b) is the perspective plot of the cleaned up Hough transform.

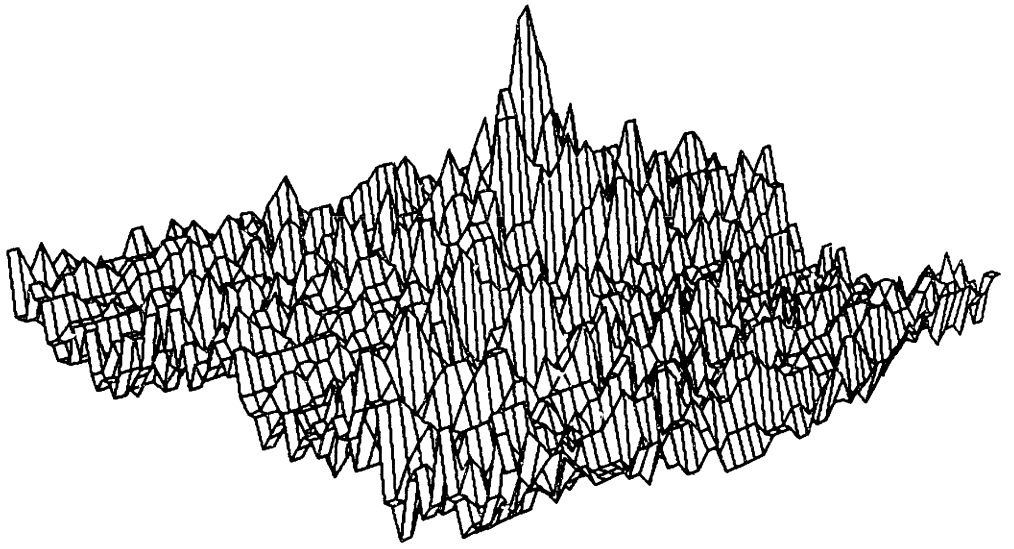


Figure 4.12(a)
Perspective plot of Hough transform in Figure 4.10(b)

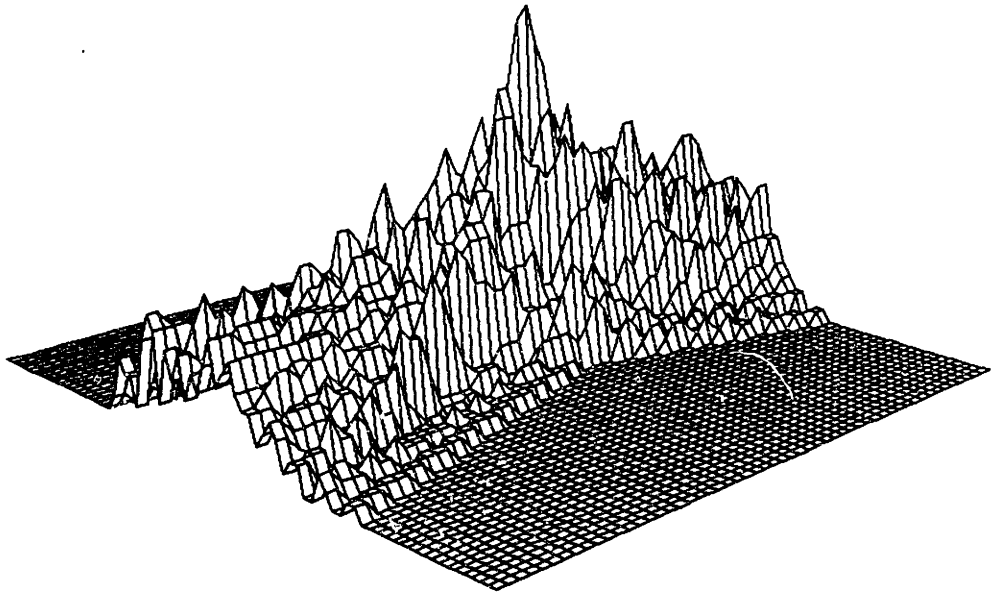


Figure 4.12(b)
Perspective plot of Hough transform in Figure 4.11(b)

Finally, Figure 4.13 shows the result of the completed curvature matching. From the Hough transform in Figure 4.10, the algorithm derived the magnification and rotation factors and, more importantly, found the end points of the matching section. Figure 4.13 shows that transform magnitudes as well as the curvatures of the resulting contours match very closely. The magnification estimate has 2 percent error and the rotation estimate has 2 degree error. However, the real merit of matching must be judged from how accurately the end points of the matching segments are found. The magnification and rotation values are global parameters that become inaccurate and eventually lose their meaning when the contours differ in shape due to noise or deformation. On the other hand, the end points depend more on the local shape of the contour and are accurate, especially near high-curvature points. The accuracy of the end points for this example was a fraction of a sample spacing.

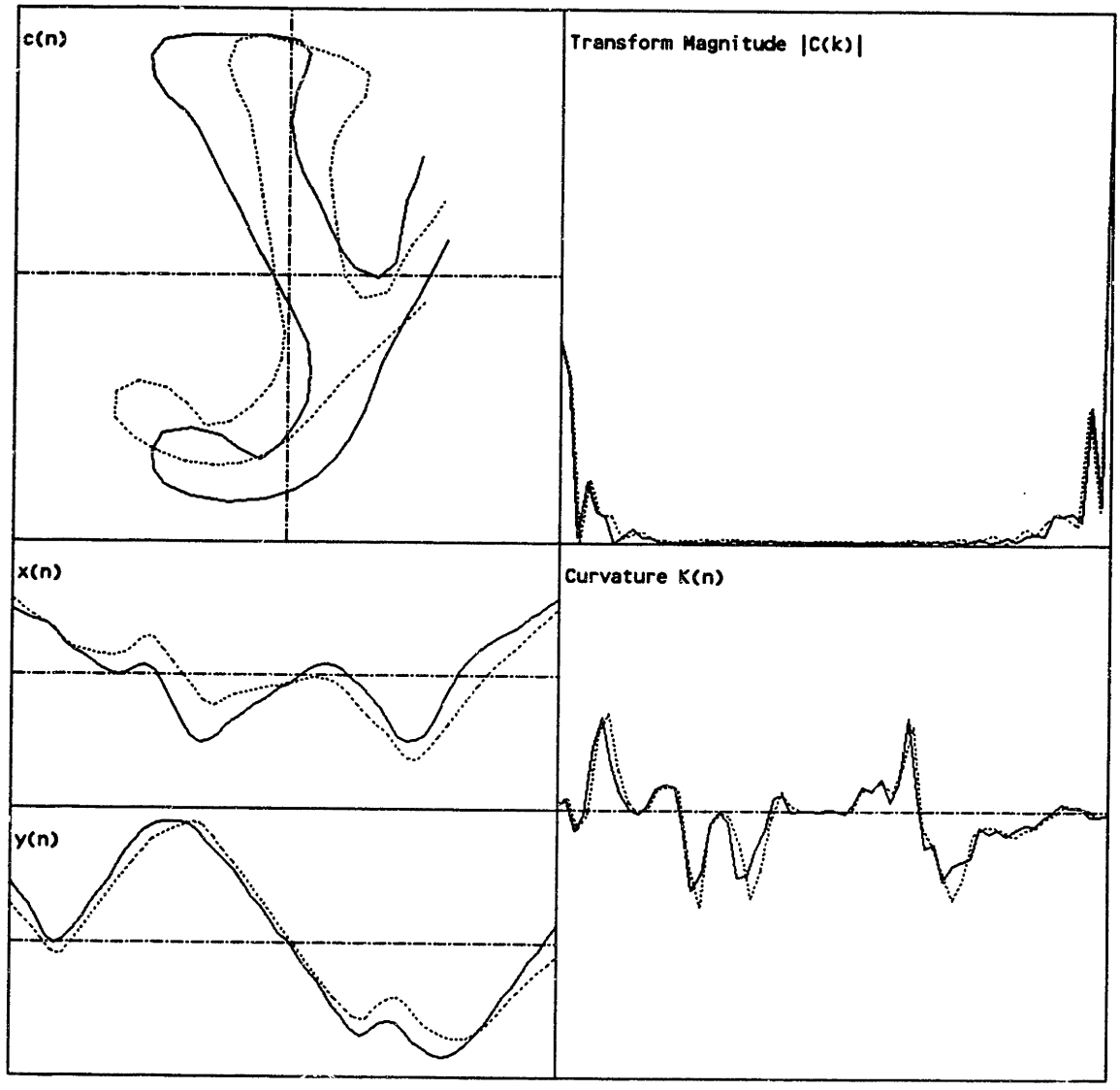


Figure 4.13
Result of the curvature matching
(Estimates for m and θ_r are 1.1296 and -0.112π)

This algorithm works reliably because of the inherent averaging that occurs in the Hough transform accumulator array. Consider a line we are looking for and the point in the Hough space to which the line maps. It is guaranteed that any other points not on this line cannot increment the accumulator cell that maps to the line. Therefore, any amount of clutter (unwanted matching curvature points) cannot affect

the peak value in the Hough space and are effectively discarded, provided that the cluttering points do not line up and form a false line on their own. Even if they form a false line and produce a false peak in the Hough space, the value for the real peak is always larger as long as the false line is shorter, and therefore contains a smaller number of points, than the line we are looking for. Basically, the matching curvature points add up *constructively* in the Hough transform space. This is essential, since our feature vector for matching is a point function (curvature) and does not average over many data points to find the feature vectors as do the transform methods. The algorithm avoids local averaging in order to attain the maximum spatial locality (as opposed to spectral locality), but instead relies on the averaging in the Hough transform space for reliable detection. This also explains why the setting of the threshold value λ is not very critical. A low threshold uniformly affects all entries in the distance matrix and reduces the number of entries that actually get mapped into the Hough space. This lowers the noise floor as well as the peak value. Of course, if the threshold is too low, very few points are mapped and the histogram becomes too sparse. Conversely, too high a threshold allows too many points to be mapped and the peak starts broadening. As long as the threshold is not in these extremes, the algorithm functions correctly because the general shape of the Hough transform does not change.

4.4 Tangent distance

If we refer to Eq. (4.2), we see that the differentiation step reduces the information content of the tangent function by one constant. That is, $T(s)$ can be recovered exactly from $K(s)$ given the constant of integration. This constant happens to be the initial tangent angle at one end of the contour. When matching two contours, each having this constant, the difference in these two constants is the rotation angle θ_r. Then there must be a warping function $W(s)$ such that the following is true (see Eq. (4.5)):

$$T_1(s) = T_2(W(s)) + \theta_r \qquad \text{for } \rho \le s \le \gamma \tag{4.19}$$

Since we assume a rigid transformation (see Eq. (4.6)),

$$T_1(s) = T_2(as + b) + \theta_r \qquad \text{for } \rho \le s \le \gamma \tag{4.20}$$

and we can find a and b as in the curvature-matching case above. For this purpose, we use the definition of the tangent function in Eq. (4.12) and let

$$D'(n_1,n_2) = T_1(n_1) - T_2(n_2) \tag{4.21}$$

which we call the *tangent difference matrix* and

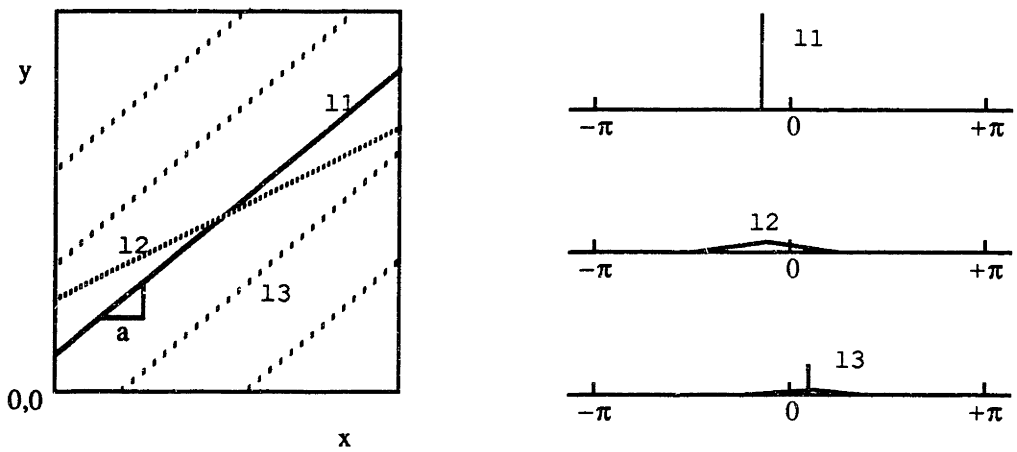
$$D(n_1,n_2) = | T_1(n_1) - T_2(n_2) - \theta_r | \tag{4.22}$$

which is the *tangent distance matrix.*

$D'(n_1,n_2)$ is a biased version of $D(n_1,n_2)$ by θ_r and $D(n_1,n_2)$ has this bias removed. In order to perform tangent matching of contours using (4.22), we need to find the rotation angle θ_r. In the curvature matching case above, this parameter was removed by a simple differentiation, and it was estimated *after* the matching. Now we need to estimate it *in order to* perform the matching.

Although this may seem like a chicken-and-egg case, a very simple method works well in finding the rotation angle. We simply take the histogram of the entries in $D'(n_1,n_2)$ and look for the histogram

peak. To see this, consider a diagonal line in $D'(n_1,n_2)$. If the diagonal line happens to correspond to the warping function $W(n)$, the entries along this line must be constant and equal to the rotation angle. These entries produce a single peak in the histogram (l1 in Figure 4.14). If the line has the wrong slope, the entries along the line have different values and their histogram spreads and does not produce a prominent peak (l2 in Figure 4.14). Only the lines with the correct slope produce peaks in the histogram. The peak value is equal to the length (number of contour points) of the matching segment, which is usually shorter than the line itself (l3 in Figure 4.14). Consider only the lines with the correct slope a. Suppose we know what this slope is and take the histogram of $D'(n_1,n_2)$ by taking entries line by line, each line with the slope a. Then the line that corresponds to the warping function must produce the highest peak since it has the longest matching segment. The entries from the non-matching portions of the segments will tend to have a flat distribution over the histogram and do not form a peak. Since it makes no difference in which order we take the entries from $D'(n_1,n_2)$ to compute the histogram, we need not know the value of a. Therefore, the location of this histogram peak is the rotation angle θ_r. (Of course this method breaks down if the assumption of rigid transformation is violated.) In order to increase the reliability, we filter the histogram before peak detection, reducing the effect of noise.



(a) Tangent difference matrix

(b) Histograms thru l1, l2, and l3

Figure 4.14

Once the rotation angle θ_r is found, we form the tangent distance matrix $D(n_1,n_2)$ as in Eq. (4.22). The rest of the procedure for finding $W(n)$ is identical to the curvature distance case. A small error in estimating θ_r is not very critical as long as it falls within the threshold λ for the distance matrix.

4.4.1 Tangent Matching Example

To illustrate tangent matching, we use the same contour pairs in Figure 4.8 that were used for the curvature matching. The lower right quadrant in Figure 4.15 shows the tangent functions for the two contours. It can be seen that if we time register the two tangent functions, there will be a fixed offset that corresponds to the rotation angle. In order to find this offset, we compute the tangent difference $D'(n_1,n_2)$, which is shown in Figure 4.16(a). The gray value 0 (dark area) maps to $-\pi$, value 128 (medium shade) maps to 0, and value 255 (light area) maps to $+\pi$ (actually 256 maps to $+\pi$ but it wraps around to 0). The histogram of $D'(n_1,n_2)$ is shown in Figure 4.17(a) and the smoothed histogram in Figure 4.17(b) (filtered by a Gaussian). The peak is found at 115, which corresponds to (115-128)/128 = -0.10156π (the correct rotation angle is -0.1π). From this rotation estimate, the tangent distance matrix $D(n_1,n_2)$ is found and shown in Figure 4.16(b) (dark = 0, light = $+\pi$). When compared to the curvature distance matrix in Figure 4.10(a), the tangent distance matrix exhibits less clutter because we have incorporated an important constraint, namely the rotation angle. The resulting Hough transform in Figure 4.16(c) also has less clutter. The threshold value used was $\lambda=0.01\pi$. To parallel the previous example for curvature matching, in Figure 4.18(a) we show the tangent distance matrix that excludes the points that do not contribute to the Hough histogram peak. The corresponding Hough transform is shown in Figure 4.18(b). The perspective plots in Figures 4.19(a) and 4.19(b) show the Hough transforms in Figure 4.16(c) and Figure 4.18(b), respectively. The values found for a and b are identical to those from the curvature matching example. Hence the matching results are the same for both methods. Finally, Figure 4.20 shows the matching result using the tangent functions in the lower right

quadrant. The curvature plots of these matching contours are identical to the curvature plots in Figure 4.13.

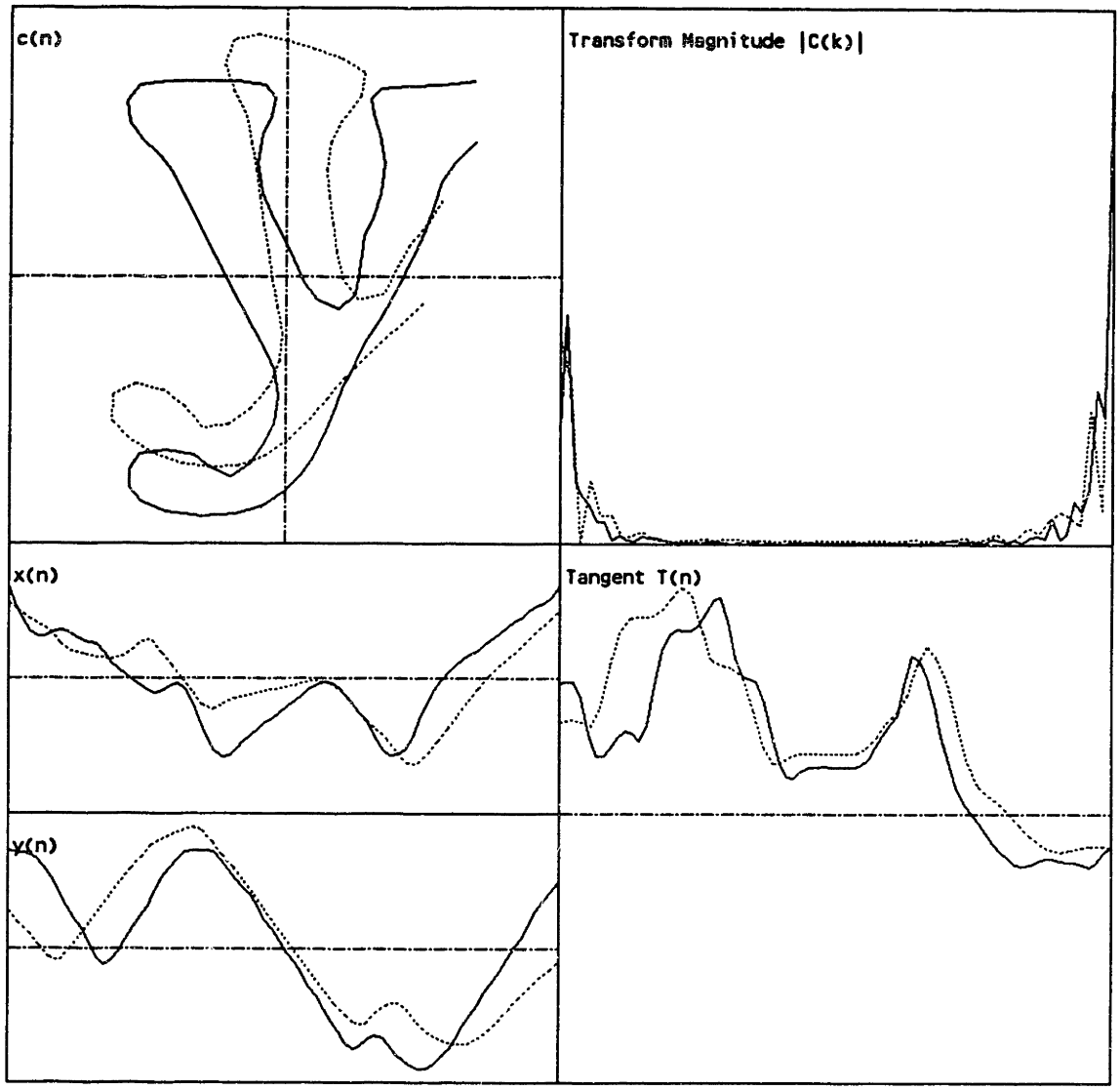
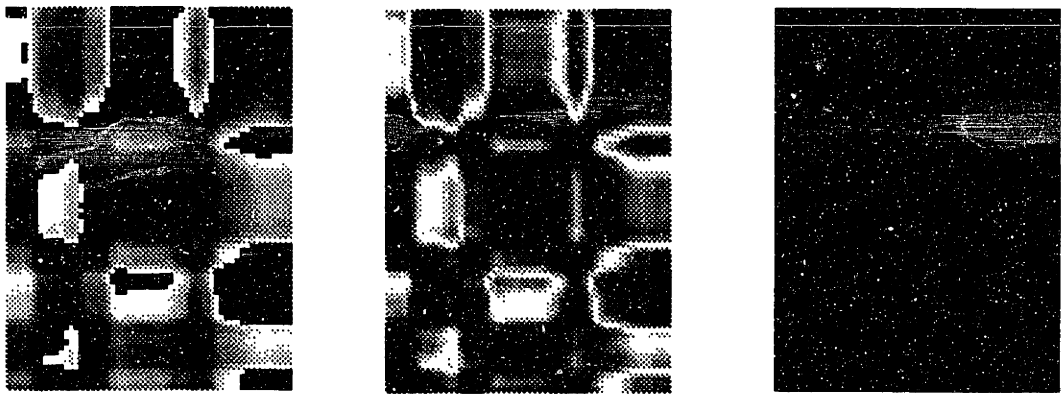


Figure 4.15
Tangent plots (lower right quadrant) for
c_1 (solid line) and c_2 (dotted line)



(a) (b) (c)

Figure 4.16
(a) Tangent difference matrix $D'(n_1,n_2)$
(b) Tangent distance matrix $D(n_1,n_2)$
(c) Hough transform of (b)

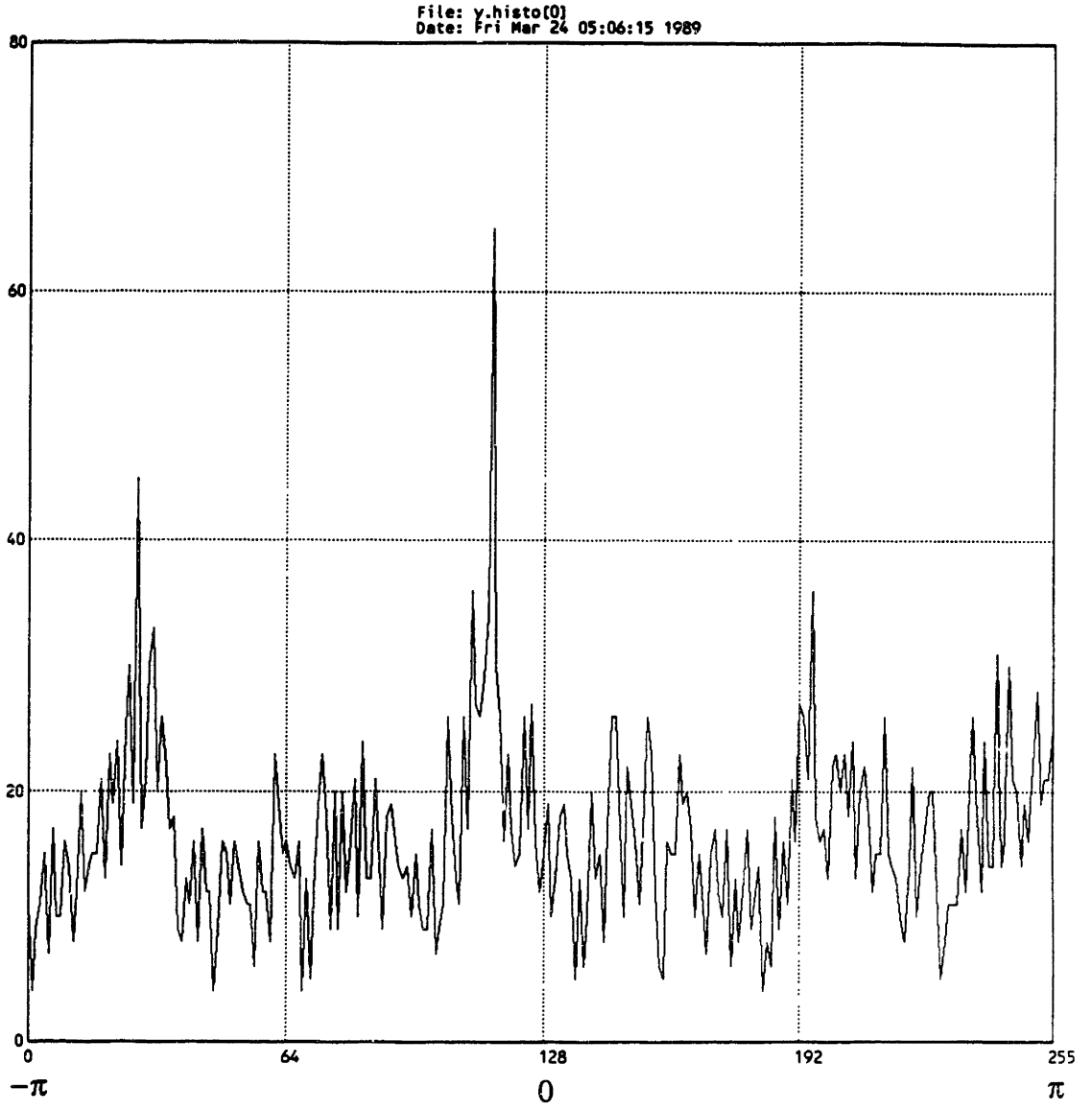


Figure 4.17(a)
Tangent difference histogram from Figure 4.16(a)

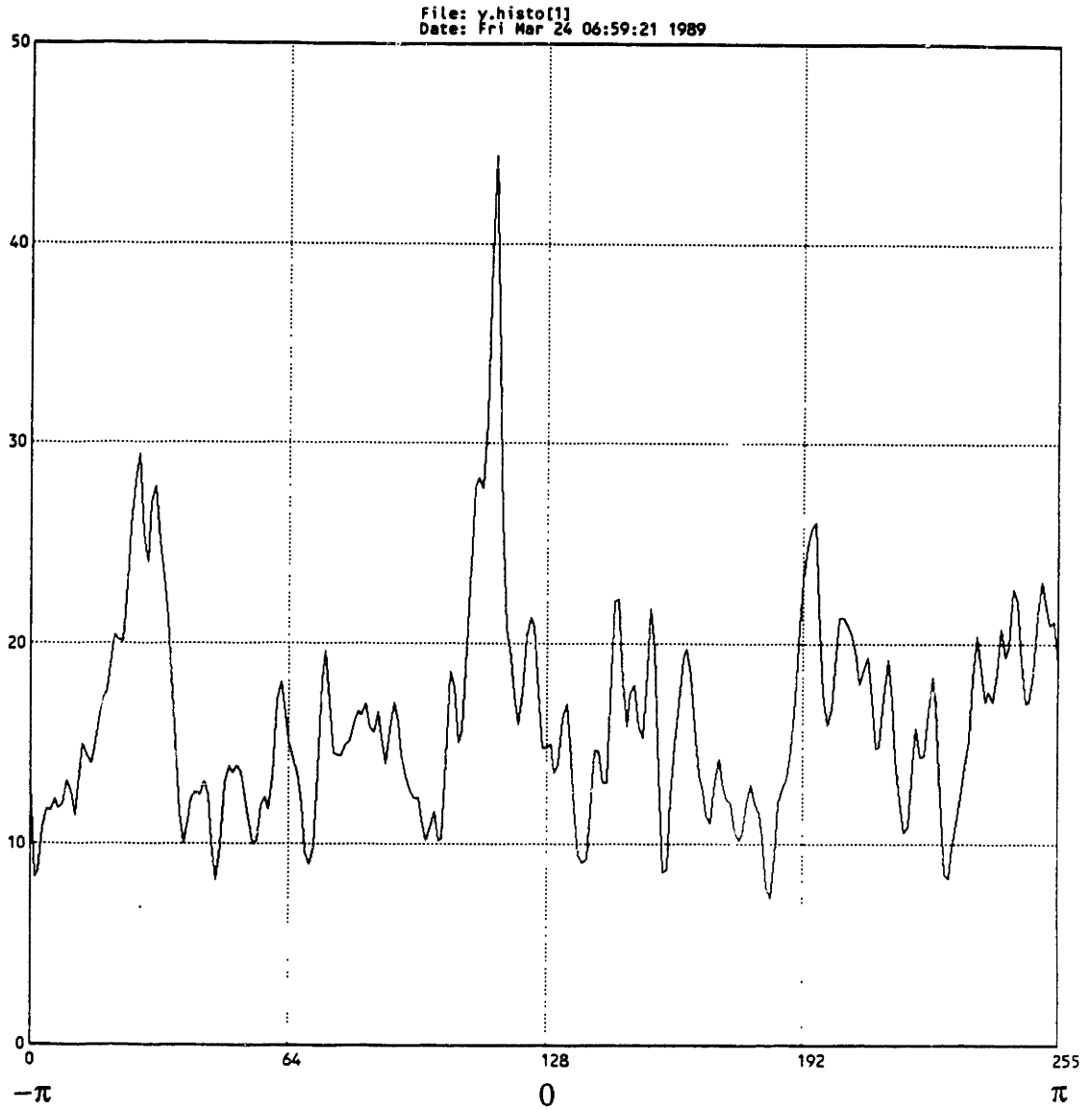
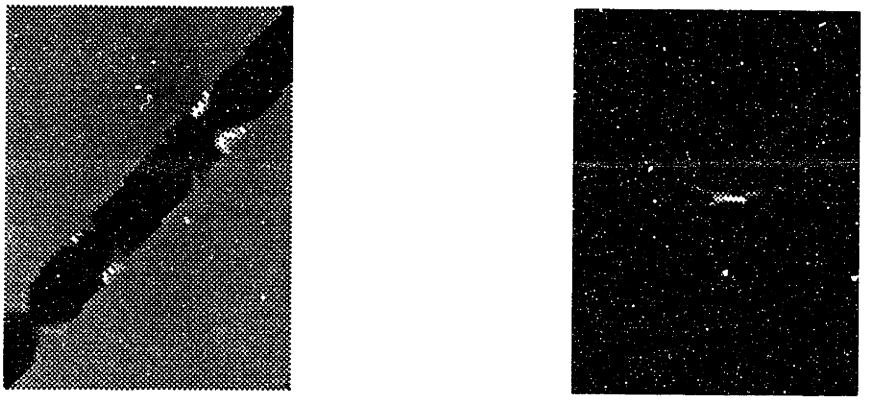


Figure 4.17(b)

Smoothed tangent difference histogram by a Gaussian filter, σ=0.5



(a) (b)

Figure 4.18
Modified curvature distance matrix (a) and its Hough transform (b)

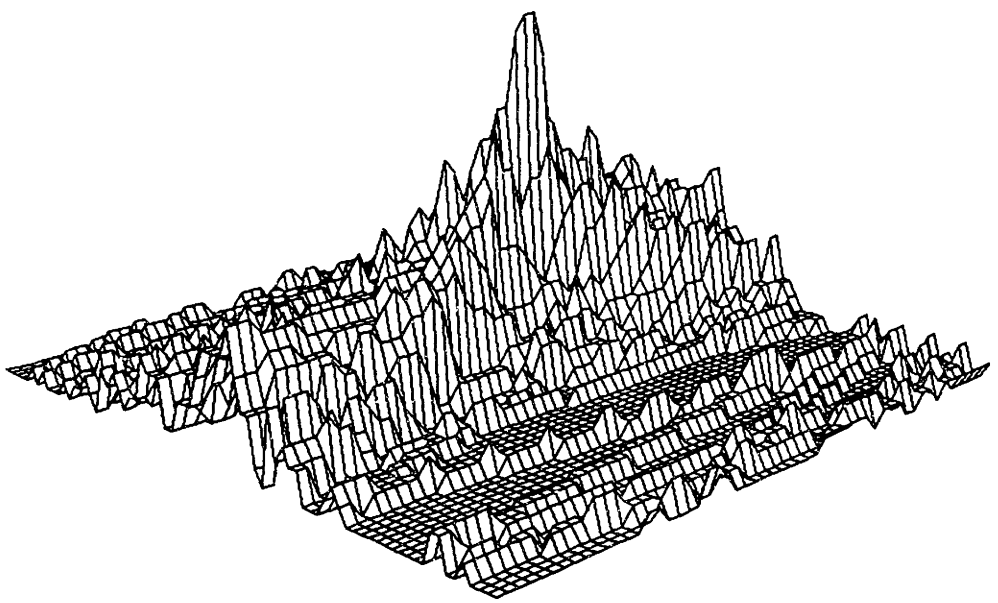


Figure 4.19(a)
Perspective plot of Hough transform in Figure 4.16(c)

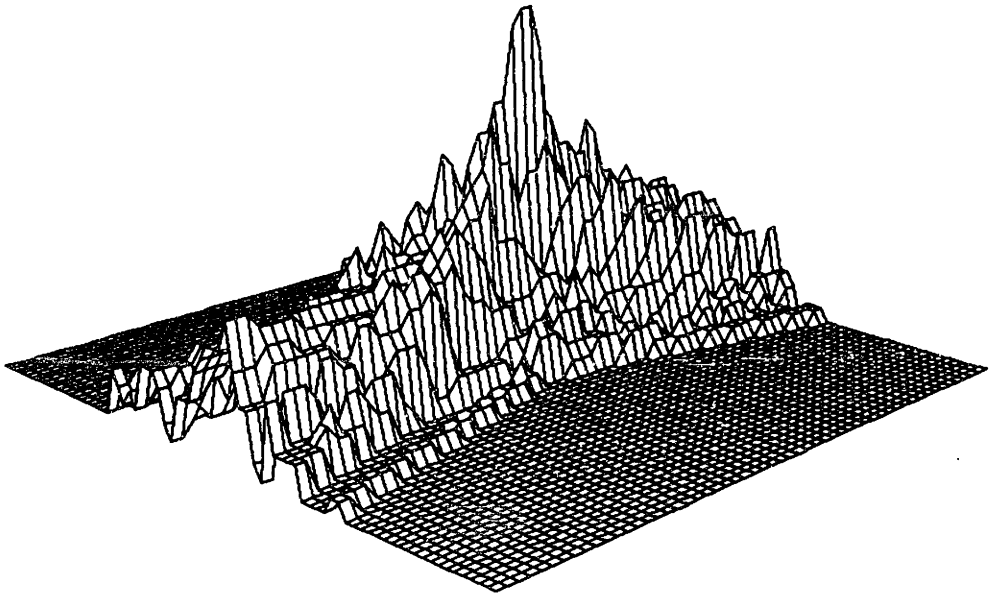


Figure 4.19(b)
Perspective plot of Hough transform in Figure 4.18(b)

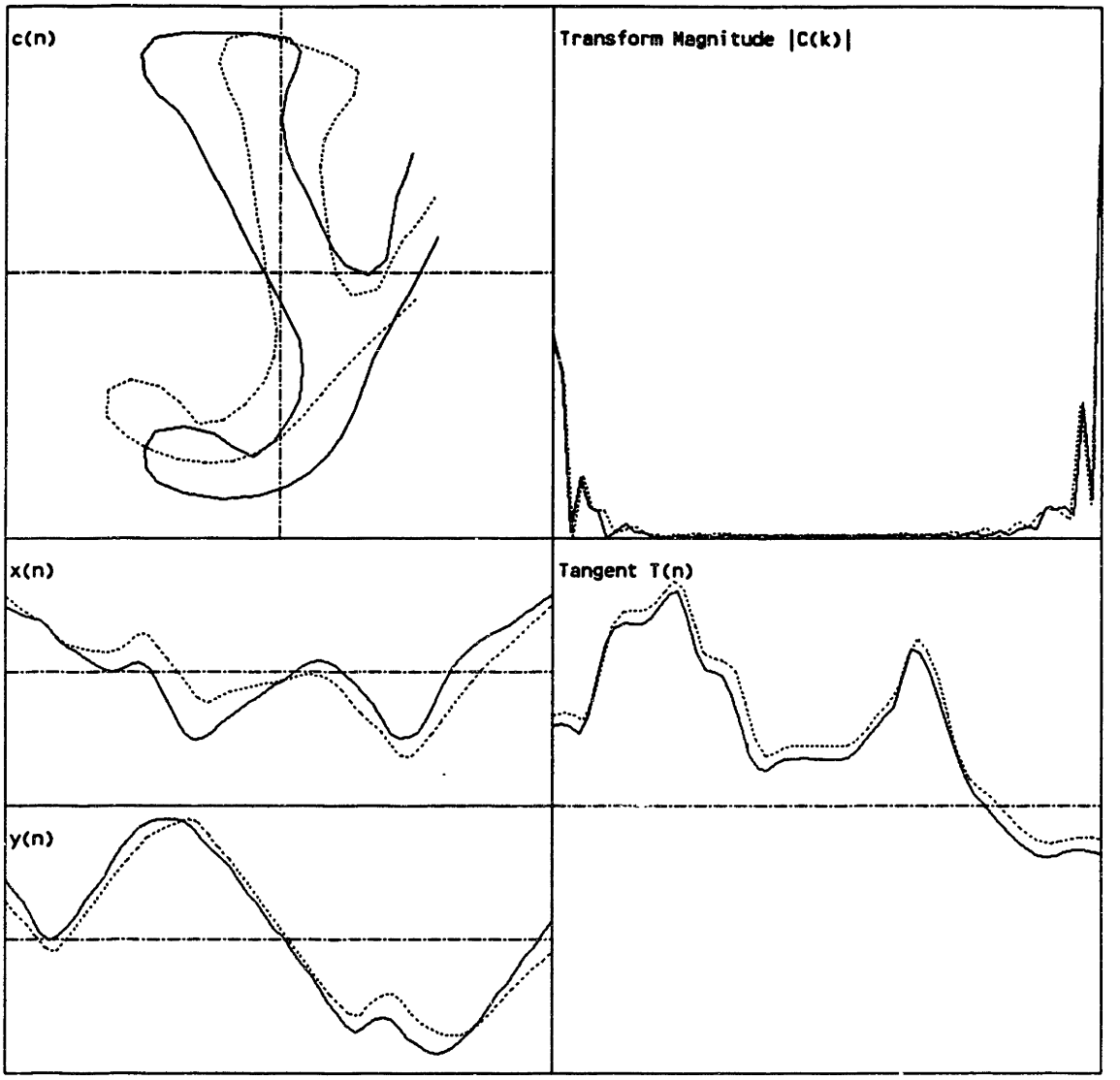


Figure 4.20
Result of the tangent matching
(Estimates for m and θ_r are 1.1296 and -0.1016π)

4.5 Interpolation

In Chapter 3, we have utilized the contour transform to interpolate between two matching closed contours in the transform domain. The interpolation was very robust and took care of many problems such as rotation, magnification, and translation, as well as minor deformation. When matching an open contour to another open contour (or to a closed contour) using the curvature or the tangent method, part of the contour is missing from the open contour. If the contours matched perfectly (matching segments are identical), then the missing segments could be copied from one another, properly transformed by the required rotation, magnification, and translation factors. However, in the presence of deformation, such surgical method does not work because of the inevitable discontinuity between the existing and added segments. In general, there is no satisfactory solution since part of the data is lost by occlusion and cannot be recovered.

4.6 Discussion

It is difficult to determine which of the two schemes would perform better in realistic circumstances. The curvature matching method avoids having to estimate the rotation before the matching. However the magnification factor m in its distance matrix (Eq. 4.16) makes one wonder about its robustness when the magnification factor is not known. The tangent matching method, on the other hand, requires the rotation angle θ_r to be known in order to compute the distance matrix (Eq. 4.22). But, a good method exists that estimates this rotation angle from the histogram of the tangent difference matrix (Figure 4.14). Both methods utilize the same Hough transform technique, so the comparison can only be made based on the computation of the distance matrix. For the sake of argument, if we let m and θ_r to be known perfectly, the only difference between the two methods is the use of the curvature function $K(n)$ versus the tangent function $T(n)$. Then the argument goes in favor of the tangent matching method because it avoids the differentiation required to compute $K(n)$ that makes the curvature method prone to noisy data. Indeed, when a moderate amount of deformation was allowed, which is usually the case when the contours are extracted from real images, the curvature method made a few incorrect matches.

Figure 4.21 shows the case where the curvature method would fail while the tangent method would find the correct matching segment. The contours were extracted from Figure 3.6(a) and 3.6(b) and represent the snap shots of a numeral 5 in motion. The occlusion at the bottom of the frame produced the open contour (dotted) which shows the top 1/3 of the closed contour (solid). Because of the transformation as well as the coarse sampling, the contour deforms moderately. Figure 4.22 shows the result of the curvature matching which failed to find the correct matching segment. Figure 4.23 shows the result of the tangent matching which found the correct match in spite of the noisy data.

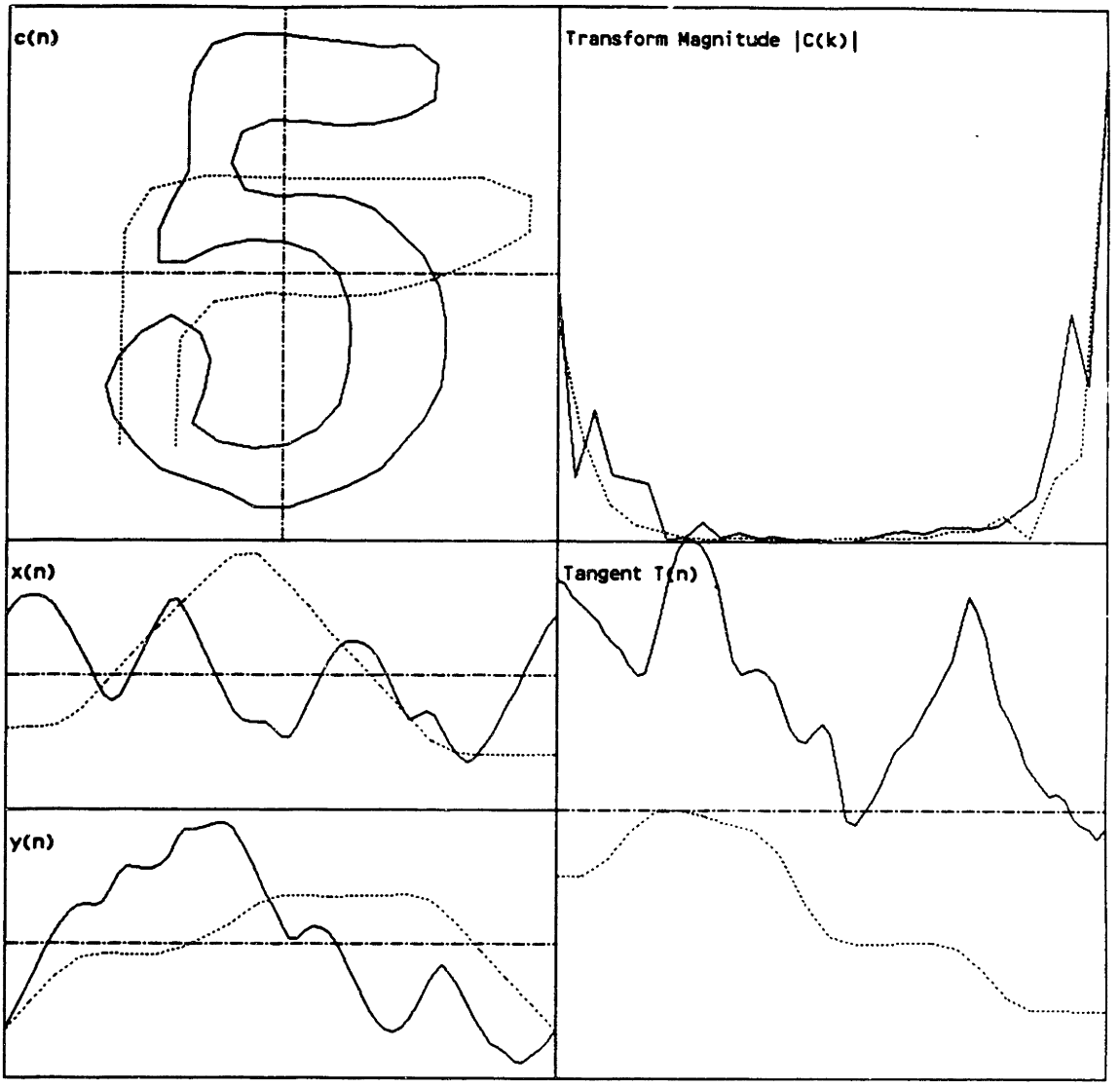


Figure 4.21
Plots for closed 5 (solid) and open 5 (dotted)
extracted from Figures 3.6(b) and 3.6(a), respectively

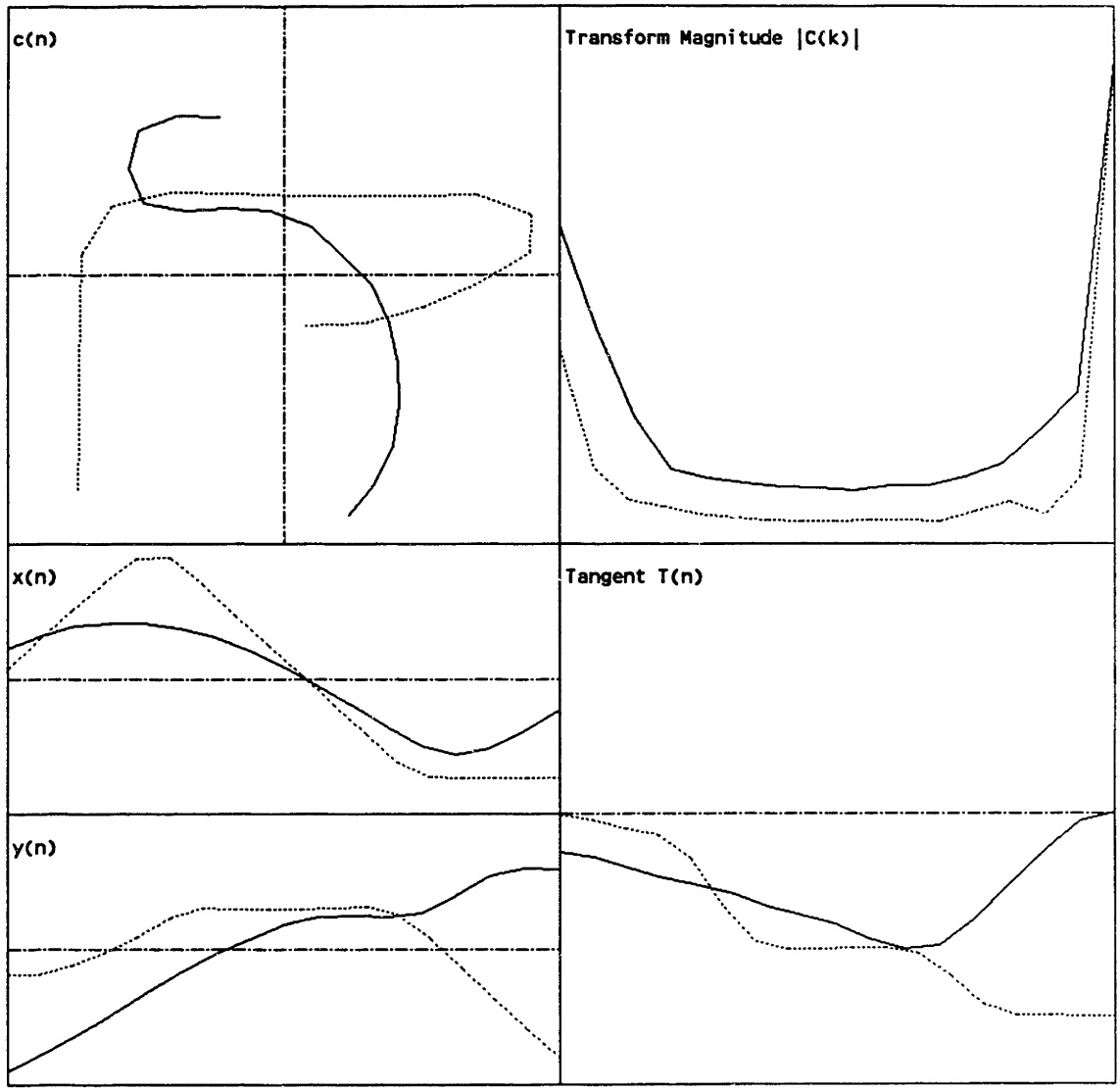


Figure 4.22
Failed curvature matching for contours in Figure 4.21

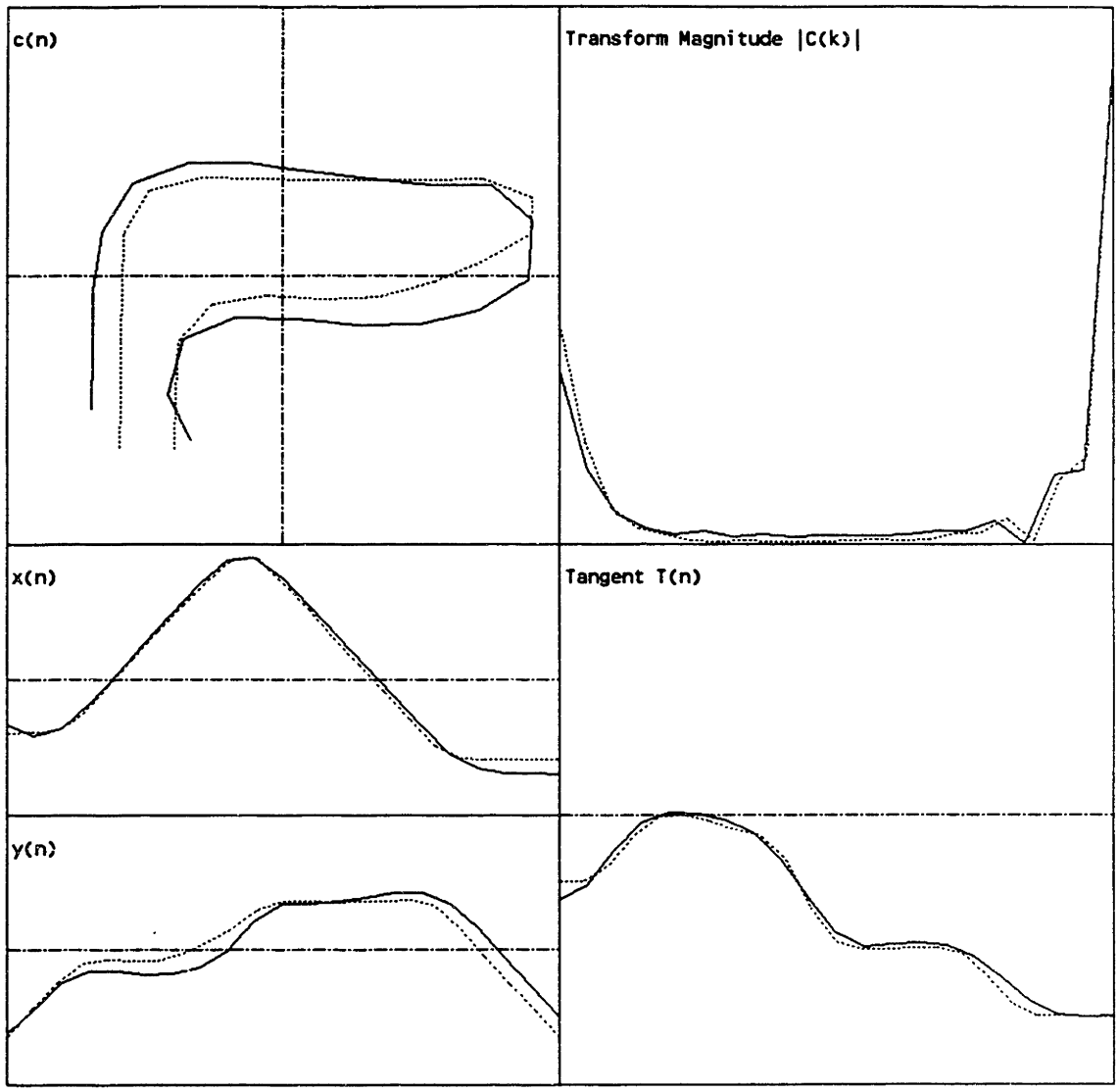


Figure 4.23
Tangent matching result of the contours in Figure 4.21

During the experiments, the tangent matching method was found to be more robust than the curvature matching method in general. However, it fails in some cases where the contour is extremely short (not enough data) or the contour is relatively featureless (ambiguous matching). Short contours do not allow enough averaging in the Hough transform space for the algorithm to work effectively. Featureless contours are the ones with constant curvature such as straight lines or

circle segments. Contours exhibiting multiple symmetry can also cause ambiguity problems. Given the obvious difficulties matching open contours in general, both algorithms performed very well.

CHAPTER 5. CONCLUSIONS

5.1 Summary

In this thesis we have taken a radically different approach to coding and motion compensating an image sequence. The traditional canonic representation of an image by a two dimensional matrix of intensity values was replaced by a region-based model of an image. This led us to represent the region boundary by contours which can be tracked over time for the purpose of motion compensation.

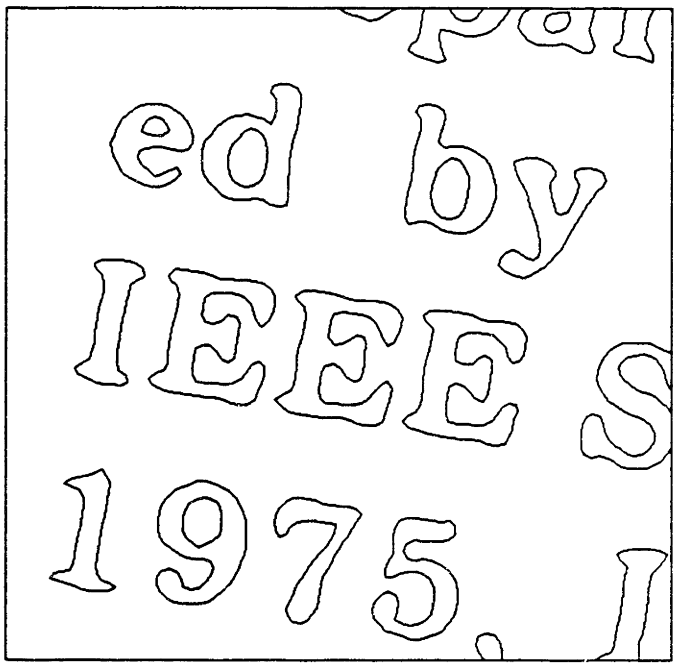
Motion compensation of images has a potential for achieving a good compression of a moving image sequence while preserving a good motion rendition. A region-based image coding can in principle achieve a very high compression by representing the image in terms of contours and textures. By combining the two techniques, a contour-texture based image sequence coding scheme can be developed. In this thesis we have elected to concentrate on the motion compensation of the moving contours. The procedure involves first finding a matching pair of contours from two successive frames, and then interpolate between the two contours.

We have developed two distinct techniques for matching contours. The first was to transform the closed contours into the frequency domain using the circular transform. Then useful transform properties were derived so that the contour can be normalized for efficient matching. The normalization procedure extracted the global shape parameters of the contour that are invariant to rotation, magnification, and translation. Then a mean squared error function was defined in the transform domain to measure the degree of similarity between normalized contours. Once the matching is established, the same shape parameters can be utilized to reconstruct the contour. To do this we developed a spatial and a temporal interpolation techniques for motion compensated interpolation of moving closed contours. The technique assumes a rigid motion but was found to be quite robust against minor deformation.

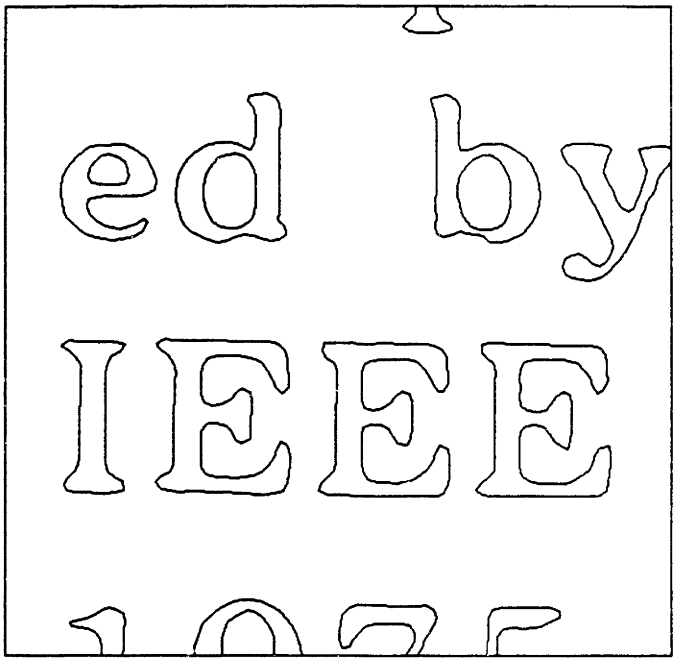
The second technique applies to open contours that arise because of occlusion. The curvature matching and tangent matching methods were developed to match and extract common matching segments of open contours. In order to maximize the spatial locality, a point function such as the curvature or the tangent along the contour is computed as the feature parameter. Then the concept of distance matrix was introduced as a way to time-scale register contours in the absence of the scale and alignment information. Assuming a rigid motion, the Hough transform was formulated to find the matching segments by estimating the scale and alignment parameters. The averaging that occurs in the Hough transform space made possible a reliable estimation of the matching parameters. Unlike the closed contour case, a good temporal interpolation is not possible between open contours since part of the data is missing due to occlusion.

5.2 Motion Compensation Example

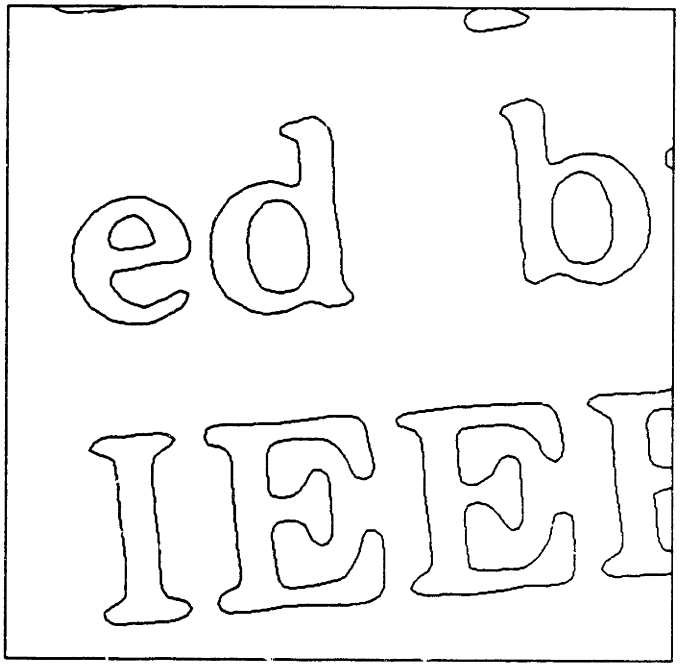
A gray scale image was scanned in at three different orientations to simulate a rigid motion. A coarse sampling (64x64) coupled with a gain variation resulted in a moderate deformation when the contours were extracted by an iso-luminance segmentation process. The three key frames are shown in Figure 5.1. Only the open contours that do not get occluded in all three frames were motion compensated using the transform technique developed in chapter 3. The up-conversion factor was 4, resulting in a 9 frame sequence that is shown in Figure 5.2 (3 extra frames between key frames). The reconstructed contours vary smoothly from one frame to the next, despite the difference in shape of the contours in the key frames.



(a)

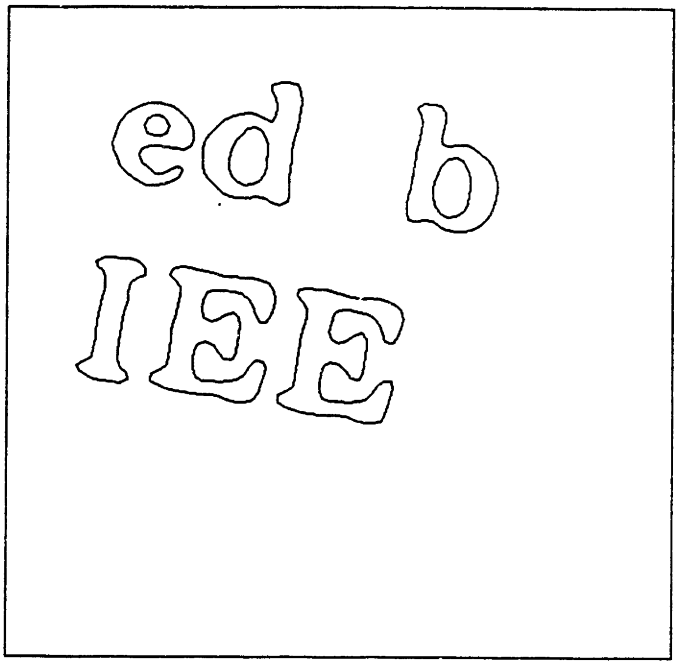


(b)

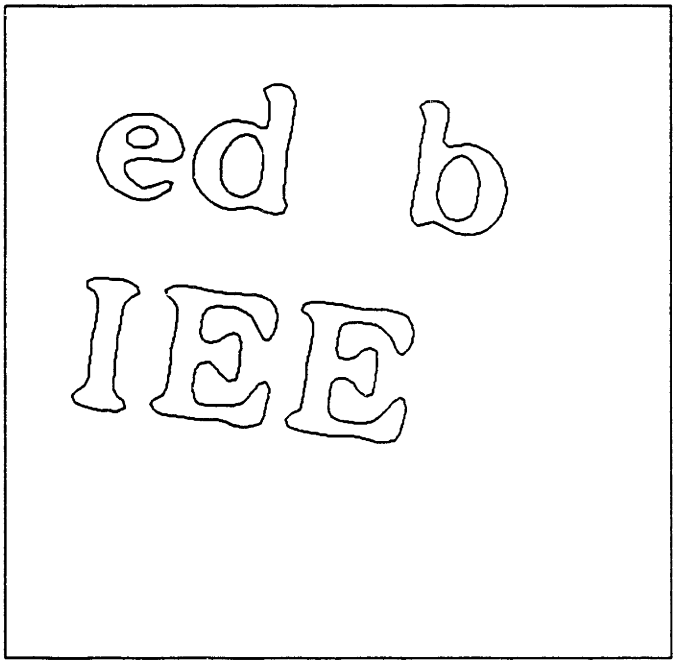


(c)

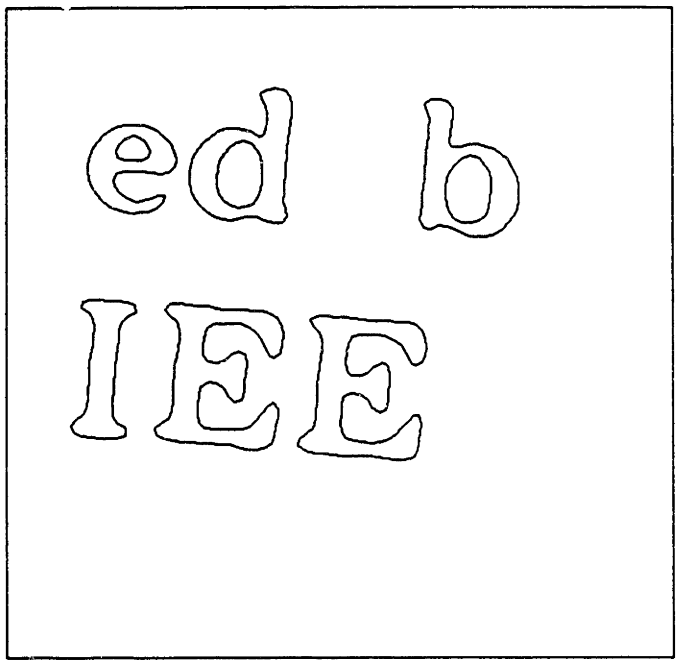
Figure 5.1
Three key frames used for motion compensation



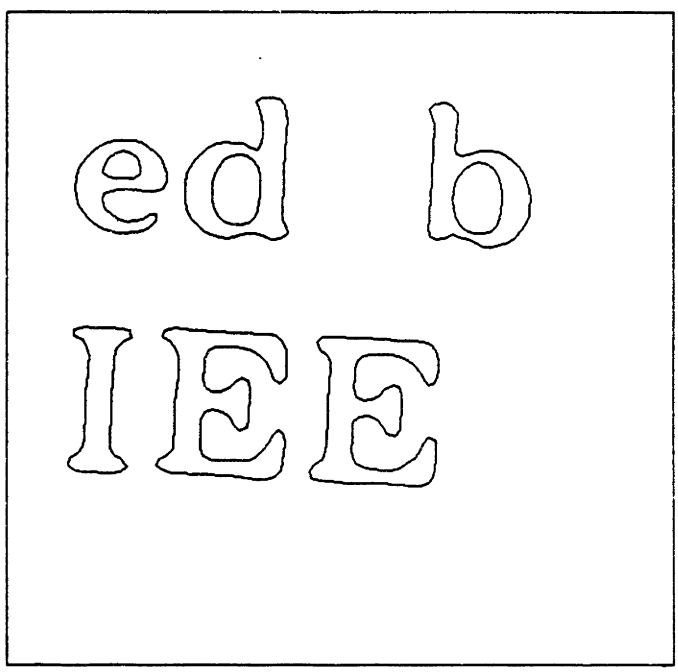
(a)



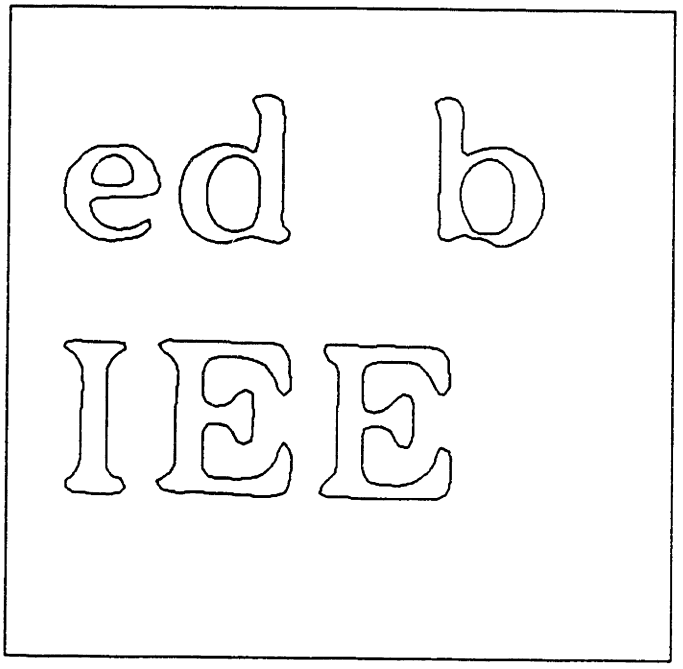
(b)



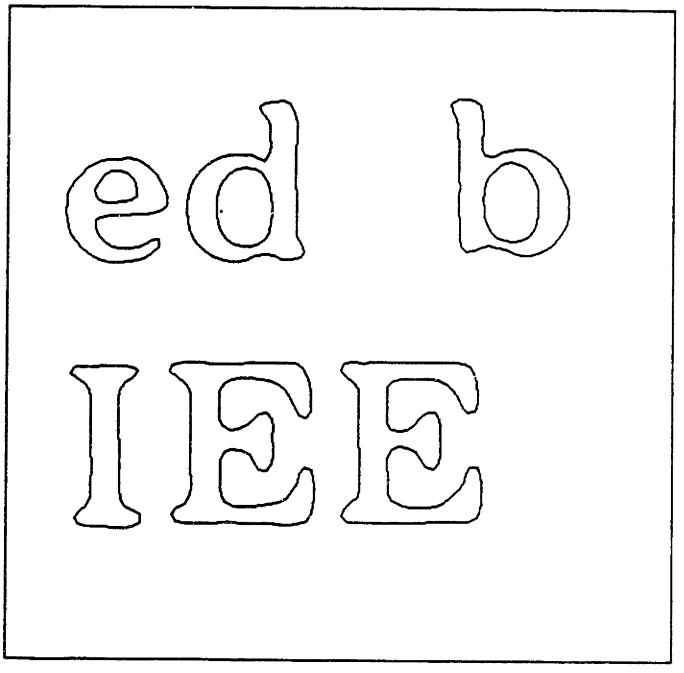
(c)



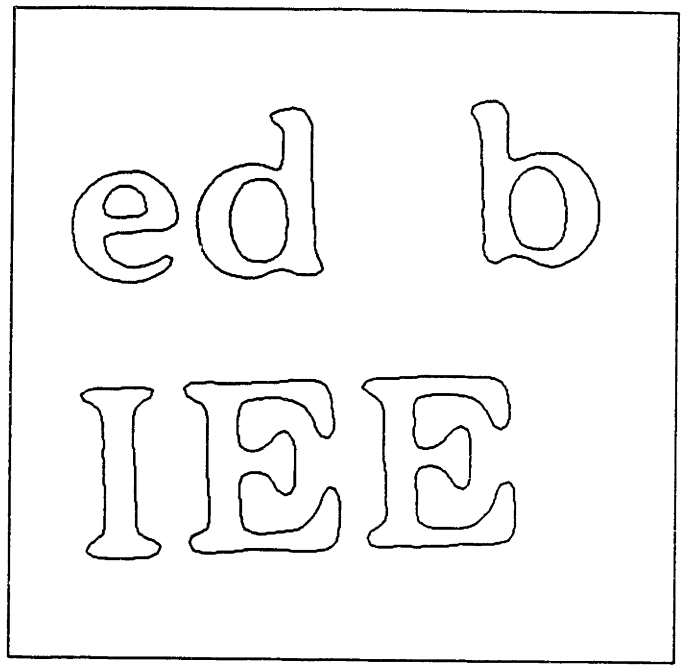
(d)



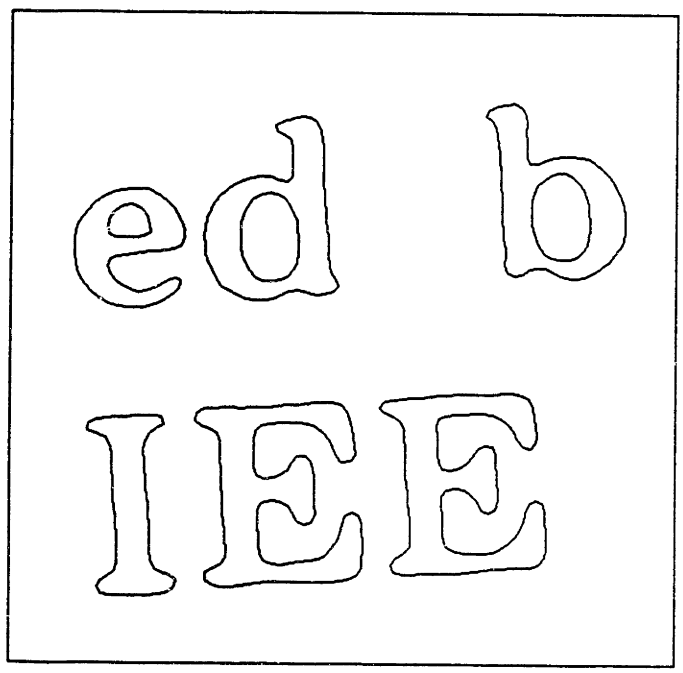
(e)



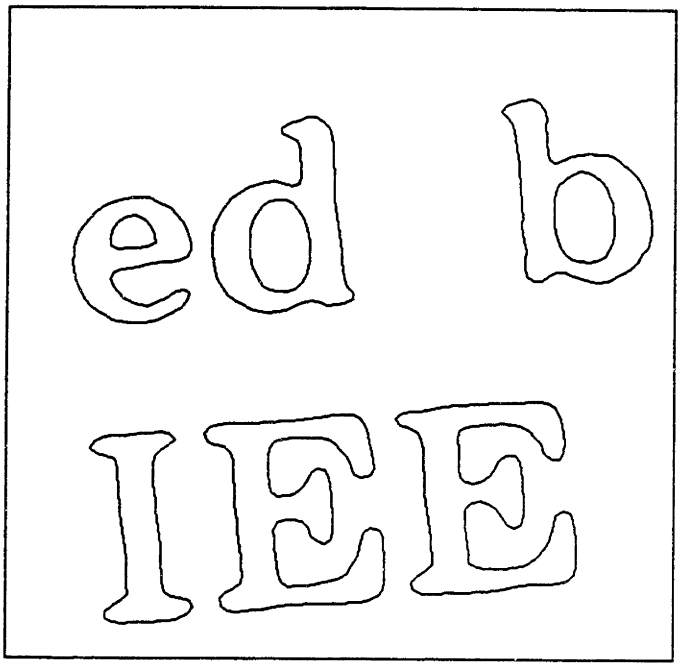
(f)



(g)



(h)



(i)

Figure 5.2

9 frames generated by motion compensation from three key frames in Figure 5.1. The interpolation factor is 4.

For an open contour example, we picked the letter E that becomes occluded at the right boundary in the third frame (Figure 5.1(c)) and the corresponding contour in the second frame (third E in Figure 5.1(b)). The open E and the closed E are shown together in Figure 5.3. Figure 5.4 shows the result of the tangent matching from which the motion information was extracted for the matching segments. Because we lack an interpolation technique for open contours, the motion parameters were applied to the closed E only. Therefore the open E was used only for motion estimation and its shape was not used for interpolation. The resulting motion compensated sequence is shown in Figure 5.5. The first 4 frames were generated using the transform technique as in Figure 5.2 and the last 5 frames were generated using the tangent matching technique.

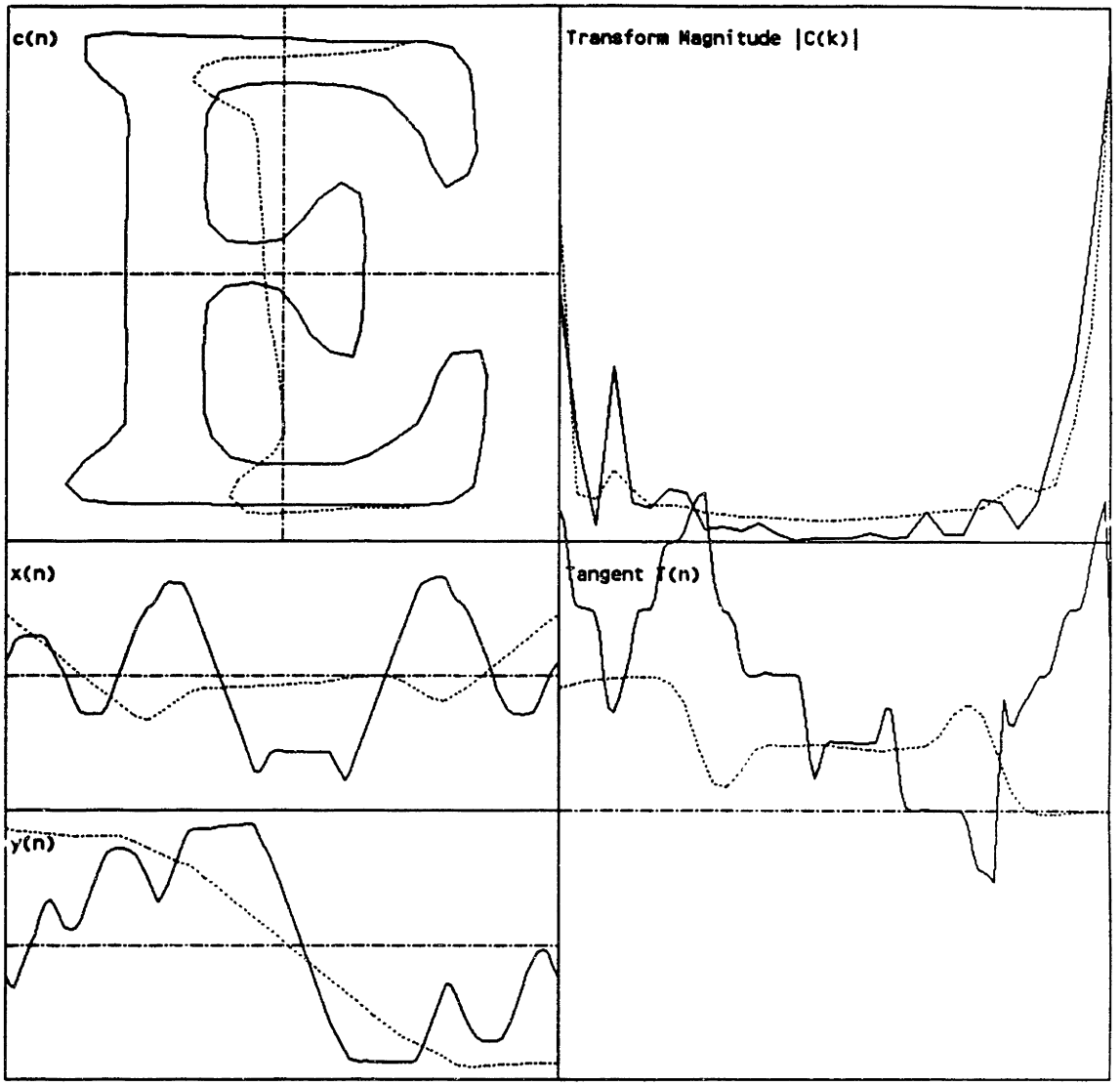


Figure 5.3
Closed E and open E from Figures 5.1(b) and 5.1(c)

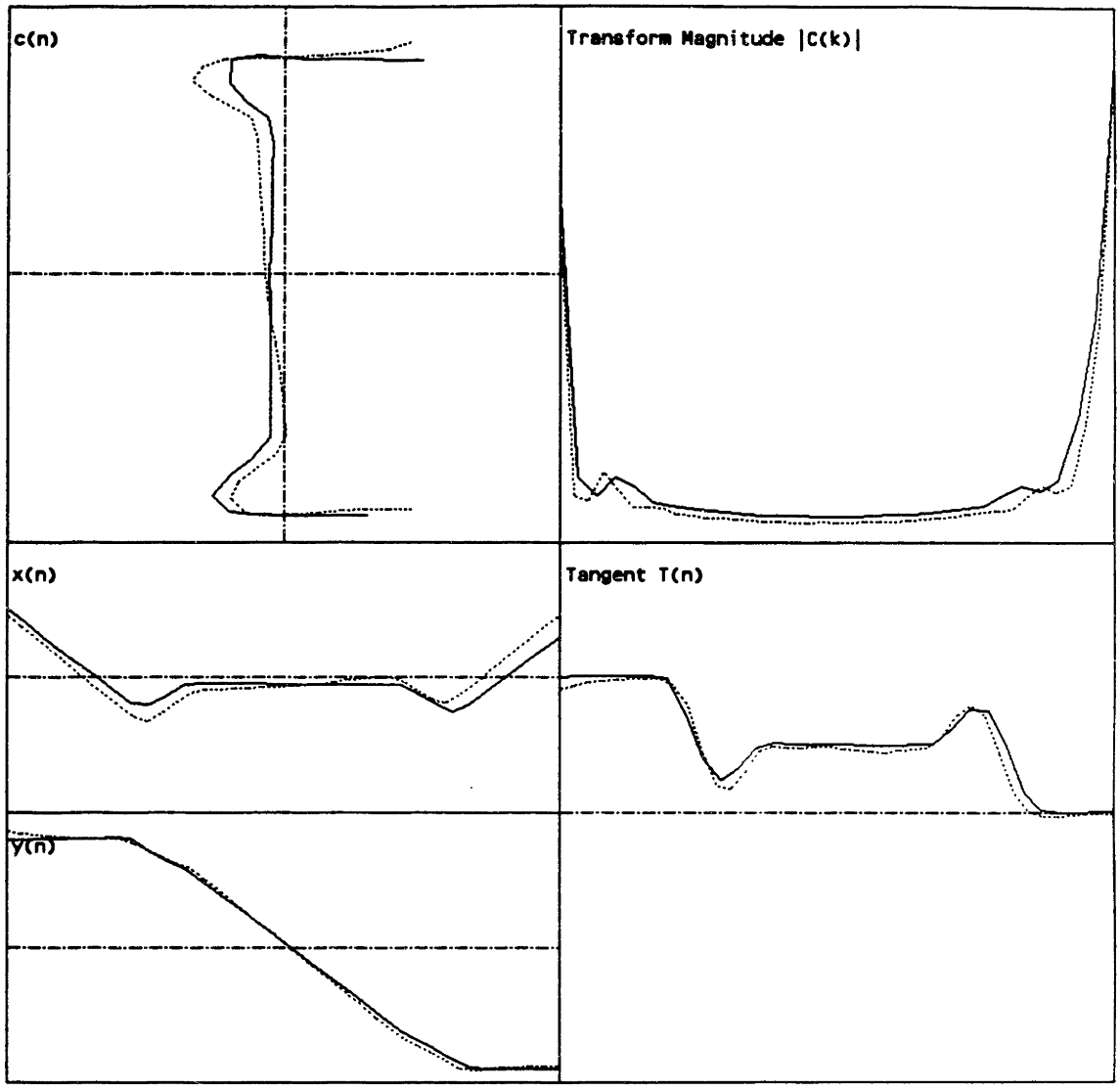
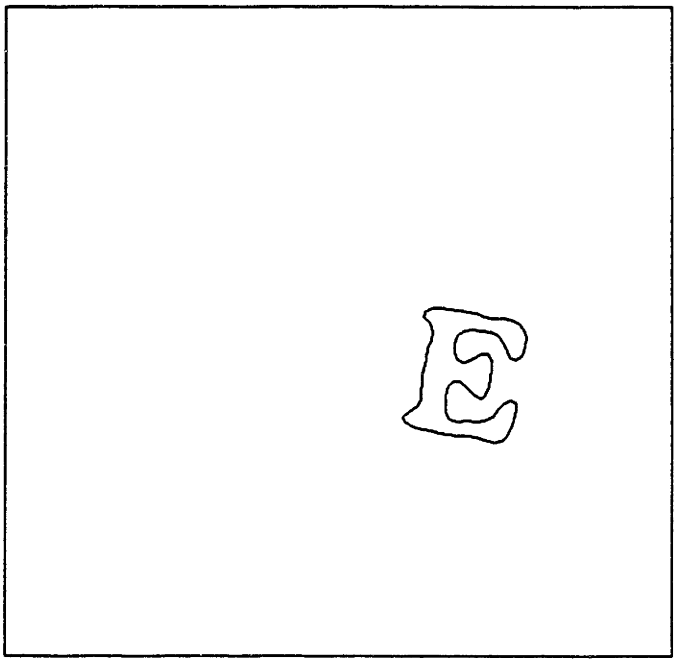
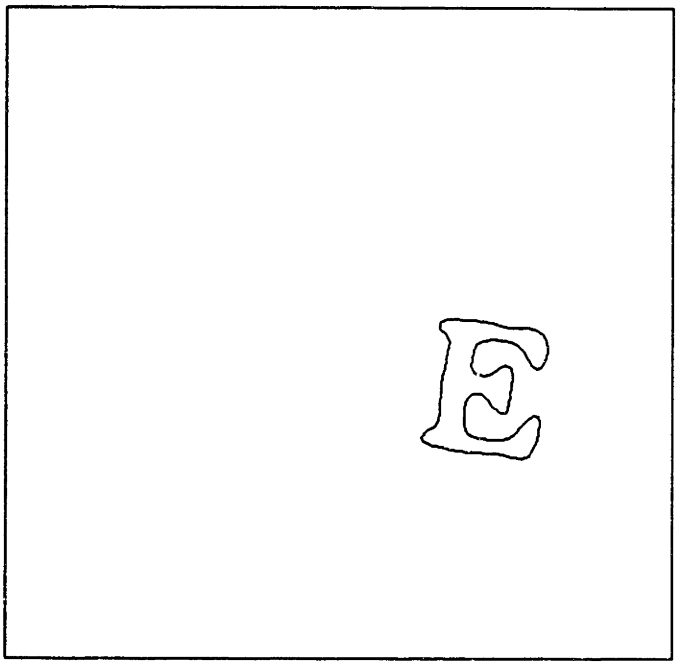


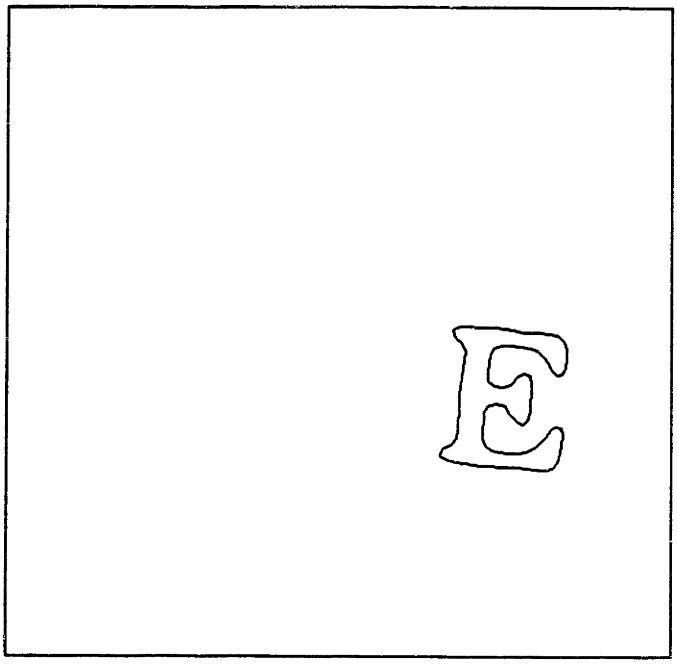
Figure 5.4
Tangent matching result of the contours in Figure 5.3



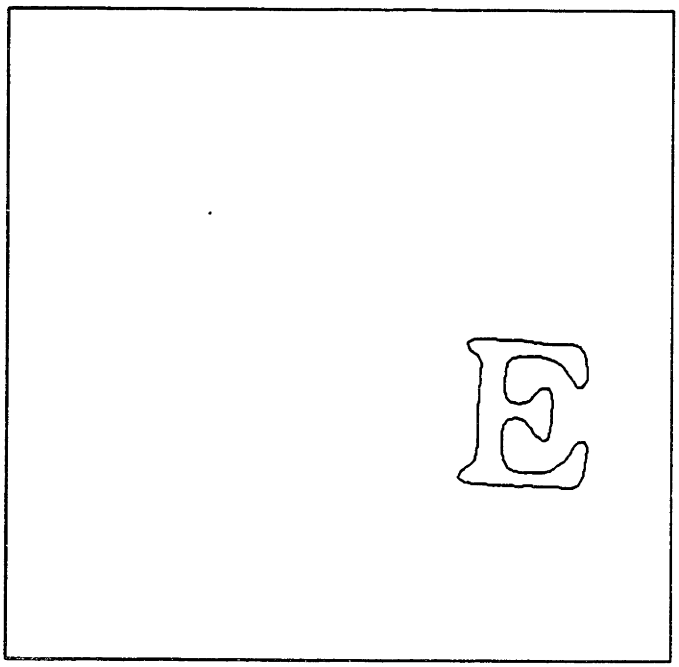
(a)



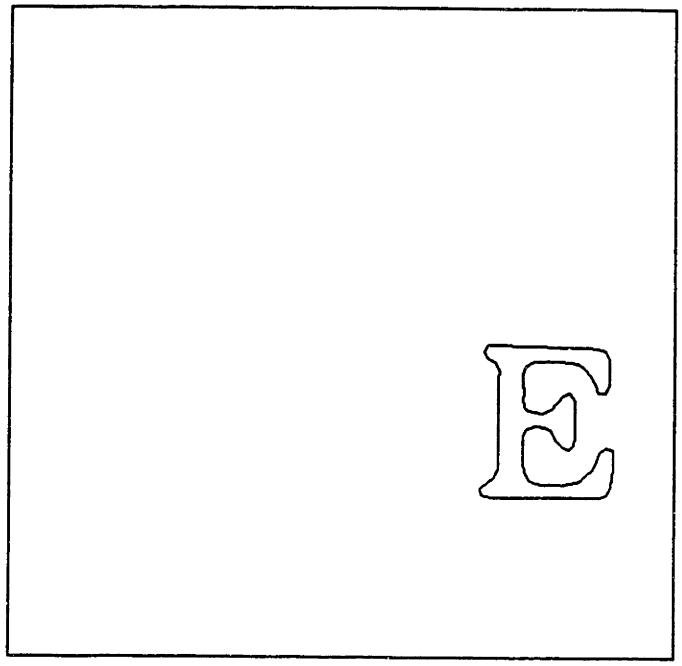
(b)



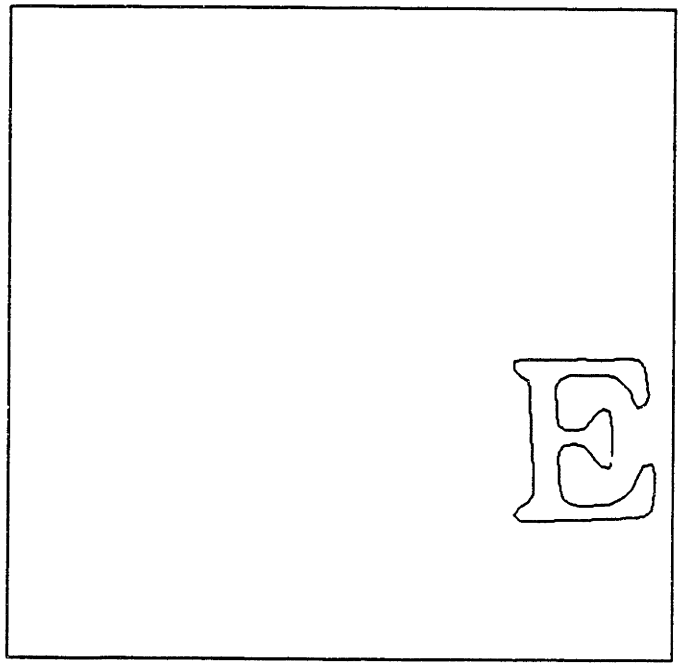
(c)



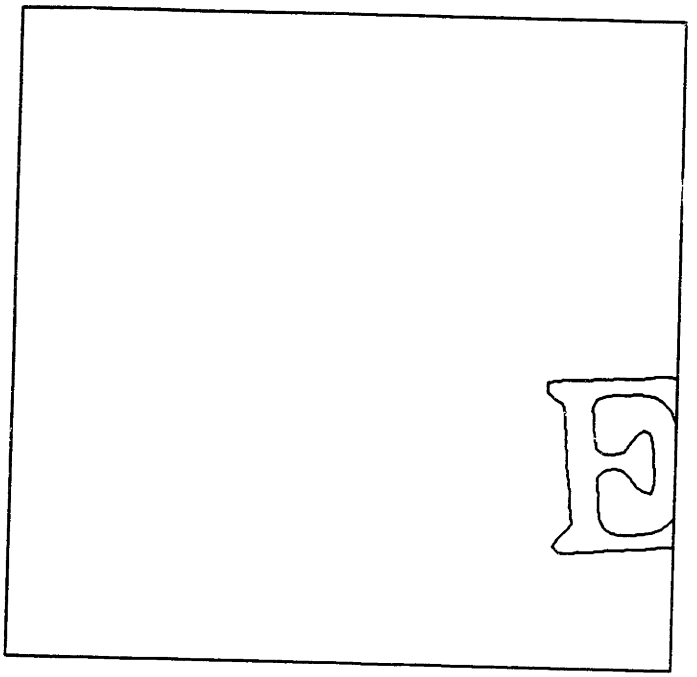
(d)



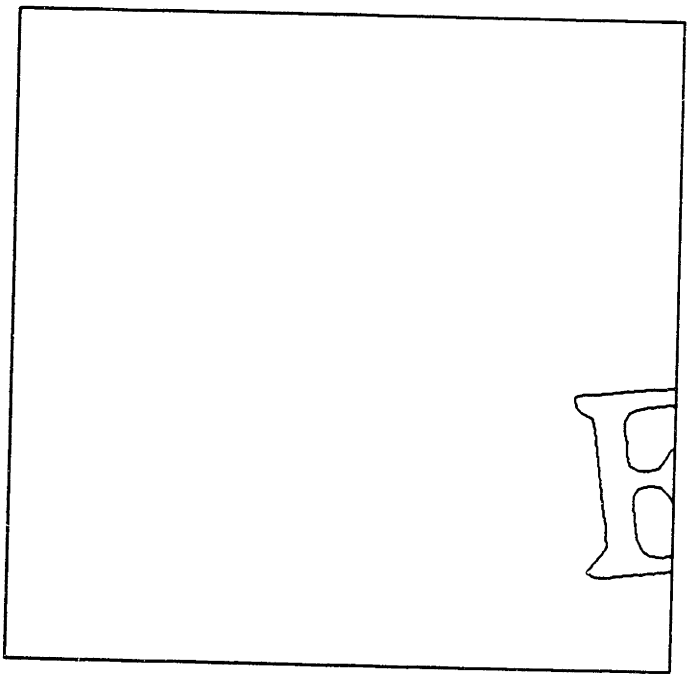
(e)



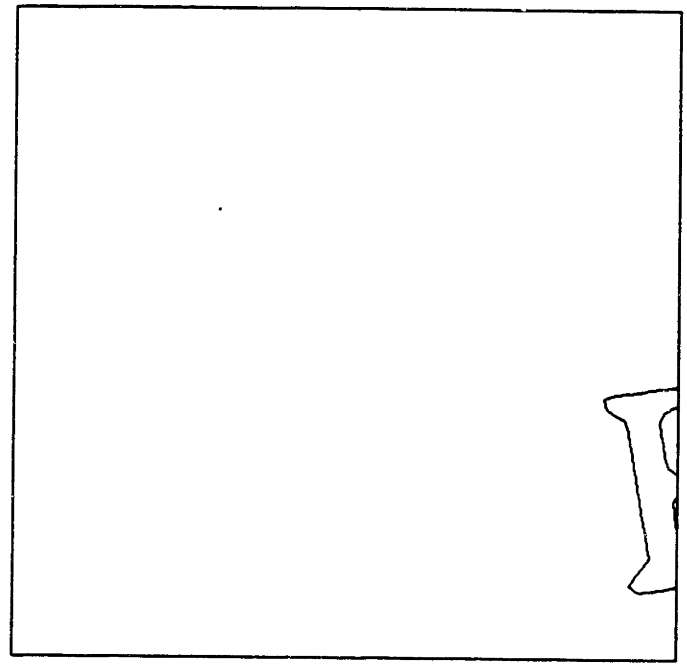
(f)



(g)



(h)



(i)

Figure 5.5

9 frames generated by motion compensation of the E that occludes at the image boundary in the third frame. The interpolation factor is 4.

5.3 Discussion

The transform technique we developed for closed contours has a good mathematical framework that is perfectly suited to the nature of closed contours. The technique resulted in a near perfect tracking and interpolation of closed contours under realistic conditions. However a few refinements are desirable. First, the normalization procedure in section 3.2.2 can be further refined to take care of any symmetry in the contour. Because the first-order ellipse examined for the normalization has a two-fold symmetry, we had to make two MSE measurements. This can be avoided if we examine the second- or higher-order ellipses. N-fold rotational symmetry is indicated by a strong presence of the (N-1)th coefficient in the contour transform. Use of higher frequency coefficients for resolving symmetry was addressed by Wallace [55]. A second refinement is needed if we desire to use more than 2 frames for temporal interpolation. In the examples, only a bilinear interpolation was performed temporally. However, a higher-order interpolation, such as a cubic spline interpolation, is straight forward using multiple frames. Because the physical law of the nature makes the occurrence of instantaneous direction changes very rare, 3 or 4 frame interpolation should be considered. Eq. 3.20 can be suitably modified to accomplish this.

The implementation of the closed contour matching is rather involved. For each contour, a complex DFT has to be computed. A FFT implementation is not possible since the length of each contour can not be made a power of 2. Also, the interpolation requires an inverse DFT. Fortunately, the number of contours in an image is relatively small, and very powerful and fast DSP chips are available that can achieve near real-time operations.

The treatment of matching open contours was somewhat less mathematical since the warping function $W(s)$ could not be solved analytically. This is because the contours are represented as discrete signals and cannot be made to be analytic functions. Instead we had to solve for an approximation to $W(s)$ by a heuristic method. In order

to simplify the procedure, we imposed a rigid-motion constraint and made $W(s)$ to be a straight line. Then the Hough transform was formulated to estimate $W(s)$. However, the Hough transform can be generalized to detect arbitrary curves that can be described by an analytic function. In principle, we can allow a non-rigid motion if the Hough transform is reformulated to detect a second- or third-order curve instead of a straight line. This would add one or two extra parameters to be estimated which requires an increase of the dimensions of the Hough transform space to 3-D or 4-D. This increase in the complexity may not be desirable in practical implementations.

In the open contour matching experiments, the end point locations were estimated to an accuracy of one sample spacing. A sub-sample accuracy can be obtained if the Hough transform space is quantized finely. An adaptive scheme [75] can be used to successively zoom-in on the region of interest to improve the accuracy without creating an excessive computational burden. However, if the contours are noisy or deformed, the rigidity assumption may not hold at such a fine scale. The alternative is to divide the matched contour segments into two smaller sub-segments. The shorter the contour segment is, the more likely it is for the rigidity assumption to hold at a fine scale. Two Hough transforms at a finer scale can be performed on the sub-segment pairs. Each transform will provide a more accurate estimate of one of the end points. The ultimate accuracy that can be obtained will depend on the noisiness of the data and the amount of curvature variation in each of the sub-segment. If the sub-segment does not have enough curvature variation (i.e. featureless), the estimate will become unreliable.

The lack of an interpolation strategy for open contours is a serious problem. In principle, there is nothing that can be done when part of the needed data is missing. The surgical method that borrows the missing segments from one another is about the only

ad-hoc alternative. The discontinuity expected may not be so bad if the deformation is small.

Although the curvature and the tangent matching techniques were developed for open contours, it seems quite possible to extend them to treat closed contours as special cases. It would involve making the distance matrix cyclic so that the warping function (a straight line) wraps around at the boundaries from top to bottom and from right to left. The Hough transform space must be made cyclic also. The b-axis can be easily made cyclic since it has a direct relationship to the y-axis of the distance matrix. However, it is not clear without further study if the a-axis needs to be made cyclic.

The implementation of the open contour matching does not require complex calculations such as DFT. The tangent calculation can be simplified using a table lookup. The curvature, if needed, can be easily derived from the tangent function. Setting up the distance matrix only requires subtraction operations. The rest of the operation involves a simple histogramming (for the rotation estimate) and a Hough transform. The Hough transform itself requires simple line drawing operations which can be implemented very efficiently [44]. Because of the broad application of the Hough transform, VLSI chips have been designed recently [77 & 78].

In general, there is no agreed method of measuring the accuracy of a motion estimation method. We have not made any effort to compare the accuracy of the contour matching method against other motion estimation techniques. Part of the difficulty arises from the fact that we estimate the motion of a contour. That is, we assign a single motion vector for each contour assuming a rigid motion. Other motion estimation techniques tries to assign a motion vector to each pixel or a small collection of pixels. Therefore we can only make a qualitative comparison.

A typical motion estimation scheme based on spatiotemporal gradient equation tries to solve for the unknown motion vector for each pixel. However, in order to reduce the effect of noise and

accommodate a long range motion, it has to adopt a some form of multi-scale approach. The image is smoothed to make a gross estimate first and the estimate is progressively refined by applying less smoothing. The accuracy of the final estimate depends on the number of pixels averaged to make the estimate. The more the averaging, the higher the potential accuracy. However, any amount of averaging basically trades off the accuracy against the spatial locality of the estimate. If too much averaging is done, the motion estimate near the motion boundary becomes inaccurate.

On the other hand, a block matching technique assumes a uniform translational motion of block of pixels. Literally, a block correlation is performed to find the motion vector for the block. Again, the larger the block size is, the more averaging it does to increase the motion accuracy, but at a reduced spatial locality of the estimate. Again, the scheme does not allow for motion discontinuities at the object boundaries such that the motion compensated reconstruction often has trouble reconstructing around the edges in the image.

On the contrary, the motion estimation techniques developed in this thesis depends on the averaging of all the samples along the contour in order to make an accurate motion estimates. The transform-domain technique averages over all the samples in the contour in order to extract the global shape parameters of the contour. The spatial-domain technique does the averaging in the Hough transform space and implicitly averages over only the matching segment of the contour. The spatial locality of the estimate is preserved because the contours *represent* the location of the motion discontinuity. In addition, the technique provides the rotation and magnification information, unlike other techniques that only provide the translational information.

The ultimate goal of this investigation was to develop a region-based image sequence coding system. We have concentrated on solving only part of the system, namely the contour motion compensation. Other pieces of the system such as the segmentation

and texture coding must be investigated before a complete coding system can be developed. It is hoped that this investigation has made us a step closer to achieving such a coding system and generate enough interest for others to join in the effort. The developed techniques may have other interesting applications in machine intelligence.

REFERENCES

[1] W. F. Schreiber, C. F. Knapp, and N. D. Kay, "Synthetic highs, an experimental TV bandwidth compression system," J. SMPTE, Vol. 68, pp. 525-537, Aug. 1959.

[2] D. N. Graham, "Image transmission by two-dimensional contour coding," Proc. IEEE, Vol. 55, No. 3, pp. 336-346, Mar. 1967.

[3] D. E. Troxel, W. F. Schreiber, et al, "A two-channel picture coding system: I -- Real-time implementation," IEEE Trans. Comm., Vol. COM-19, No. 12, Dec. 1981.

[4] W. F. Schreiber, "The mathematical foundation of the synthetic highs system," MIT, RLE Quart. Progr. Rep., No. 68, pp. 140-146, Jan. 1963.

[5] W. F. Schreiber, "Texture in Contour-coded pictures," MIT RLE Internal Memo, Nov. 13, 1967.

[6] W. F. Schreiber, T. S. Huang, and O. J. Tretiak, "Contour coding of images," Wescon 1968.

[7] M. Kunt, A. Ikonomopoulos, and M. Kocher, "Second-generation image-coding techniques," Proc. IEEE, Vol. 73, No. 4, pp. 549-574, 1985.

[8] S. Ullman, "Analysis of visual motion by biological and computer systems," Computer 14, pp. 57-69, 1981.

[9] C. A. Darmon, "Recursive method to apply the Hough transform to a set of moving objects," Proc. ICASSP 82, pp. 825-829, 1982.

[10] L. S. Davis, Z. Wu, and H. Sun, "Contour-based motion estimation," Comput. Vision, Graph., Imag. Process. 23, pp. 313-326, 1983.

[11] E. C. Hildreth and S. Ullman, "The measurement of visual motion," MIT A.I. Memo No. 699, Dec. 1982.

[12] E. C. Hildreth, "The computation of the velocity field," MIT A.I. Memo No. 734, Sep. 1983.

[13] A. L. Yuille, "The smoothest velocity field and token matching schemes," MIT A.I. Memo No. 724, Aug. 1983.

[14] S. Shafer and T. Kanade, "Recursive region segmentation by analysis of histograms," Proc. ICASSP 82, pp. 1166-1171., 1982.

[15] Y. Ohta, T. Kanade, and T. Sakai, "Color information for region segmentation," Comput. Graph. Image Process 13, pp. 222-241, 1980.

[16] G. B. Coleman and H. C. Andrews, "Image segmentation by clustering," Proc. IEEE, Vol. 67, No. 5, pp. 773-785, May 1979.

[17] M. Kocher and M. Kunt, "A contour-texture approach to picture coding," Proc. ICASSP 82, pp. 436-439, 1982.

[18] J. O. Eklundh, H. Yamamoto, and A. Rosenfeld, "A relaxation method for multispectral pixel classification," IEEE. Trans. Pattern. Anal. Machine Intell., Vol. PAMI-2, No. 1, pp. 72-75, Jan. 1980.

[19] A. R. Hanson and E. M. Riseman, "Segmentation of natural scenes," in Computer Vision Systems, A. Hanson and E. Riseman Eds., Academic Press, New York, 1978, pp. 129-163.

[20] W. Rutkowski, S. Peleg, and A. Rosenfeld, "Shape segmentation using relaxation," IEEE. Trans. Pattern. Anal. Machine Intell., Vol. PAMI-3, No. 4, pp. 368-375, July 1981.

[21] L. S. Davis, "A survey of edge detection techniques," Comput. Graph. Image Process. 4, pp. 248-270, 1975.

[22] M. Kunt, "Edge detection: A Tutorial review," Proc. ICASSP 82, pp. 1172-1175, 1982.

[23] I. E. Abdou and W. K. Pratt, "Quantitative design and evaluation of enhancement/thresholding edge detectors," Proc. IEEE, Vol. 67, No. 5, pp. 753-763, May 1979.

[24] W. B. Thompson, K. M. Mutch, and V. Berzins, "Edge detection in optical flow fields," Proc. 2nd Nat. Conf. Artif. Intell., pp. 26-29, Pittsburg, PA, Aug. 1982,

[25] T. Pavlidis, "A thinning algorithm for discrete binary images," Comput. Graph. Image Process. 13, pp. 142-157, 1980.

[26] M. Basseville, "Edge detection using sequential methods for change in level -- Part II: Sequential detection of change in

mean," IEEE Trans. Acoust., Speech, Signal Process., Vol. ASSP-29, No. 1, pp. 32-50, Feb. 1981.

[27] W. A. Perkins, "Area segmentation of images using edge points," IEEE Trans. Pattern Anal. Machine Intell., Vol. PAMI-2, No. 1, pp. 8-15, Jan. 1980.

[28] J. M. Prager, "Extracting and labeling boundary segments in natural scenes," IEEE Trans. Pattern Anal. Machine Intell., Vol. PAMI-2, No. 1, pp. 16-27, Jan. 1980.

[29] H. Derbin and H. Elliot, "Modeling and segmentation of noisy and textured images using Gibbs random fields," IEEE Trans. Pattern Anal. Machine Intell., Vol. PAMI-9, No. 1, pp. 39-55, Jan. 1987.

[30] S. K. Pal and R. A. King, "On edge detection of X-ray images using fuzzy sets," IEEE Trans. Pattern Anal. Machine Intell., Vol. PAMI-5, No. 1, pp. 69-77, Jan. 1983.

[31] C. W. K. Gritton and E. A. Parrish, Jr., "Boundary location from an initial plan: The bead chain algorithm," "On edge detection of X-ray images using fuzzy sets," IEEE Trans. Pattern Anal. Machine Intell., Vol. PAMI-5, No. 1, pp. 8-13, Jan. 1983.

[32] A. Rosenfeld and L. S. Davis, "Image segmentation and image models," Proc. IEEE, Vol. 67, No. 5, pp. 764-772, May 1979.

[33] M. A. Fischler, S. T. Barnard, R. C. Bolles, M. Lowry, L. Quam, G. Smith, and A. Witkin, "Modeling and using physical constraints in scene analysis," Proc. 2nd Nat. Conf. Artif. Intell., pp. 30-35, Pittsburg, PA, Aug. 1982,

[34] P. Fah and M. Kunt, "Efficient coding of high resolution typographical characters," Proc. ICASSP 82, pp. 440-443, 1982.

[35] U. Ramer, "An iterative procedure for the polygonal approximation of plane curves," Comp. Graphics and Image Proc. 1, pp. 244-256, 1972.

[36] J. G. Dunham, "Optimum uniform piecewise linear approximation of planar curves," IEEE Trans. Pattern Anal. Machine Intell., Vol. PAMI-8, No. 1, pp. 67-75, Jan. 1986.

[37] H. Freeman, "On encoding of arbitrary geometric configurations," IRE Trans. Elect. Computers, EC-10, pp. 260-268, June 1961.

[38] J. Koplowitz, "A unified theory of coding schemes for the efficient transmission of line drawings," Proc. Canadian Conf. on Commun. and Power, pp. 205-208, Oct. 1976.

[39] H. Freeman, "A review of relevant problems in the processing of line-drawing data," in Automatic Identification and Classification of Images, Grasselli, Ed., New York: Academic Press, 1969, pp. 155-174.

[40] H. Freeman, "Computer processing of line-drawing images," Computing Surveys, Vol. 6, No. 1, pp. 57-97, Mar. 1974.

[41] P. Zamperoni, "Scaling of contour-coded binary images," Electronic Letters, Vol. 14, No. 19, pp. 608-610, 14th Sept. 1978.

[42] J. Koplowitz, "On the performance of chain codes for quantization of line drawings," IEEE Trans. Pattern Anal. Machine Intell., Vol. PAMI-3, No. 2, pp. 180-185, Mar. 1981.

[43] D. Neuhoff and K. Castor, "A rate and distortion analysis of chain codes for line drawings," IEEE Trans. on Info. Theory, Vol. IT-31, No. 1, pp. 53-68, Jan. 1985.

[44] J. E. Bresenham, "Algorithm for computer control of a digital plotter," IBM Systems Journal, Vol. 4, No. 1, pp. 25-30, 1965.

[45] R. D. Merill, "Representation of contours and regions for efficient computer search," Commun. of the ACM, Vol. 16, No. 2, pp. 69-82, Feb. 1973.

[46] G. Sidhu and R. Boute, "Property encoding: Application in binary picture encoding and boundary following," IEEE Trans. on Computers, Vol. c-21, No. 11, pp. 1206-1216, Nov. 1972.

[47] J. Loeb, "Communication theory of transmission of simple drawings," Proc. London Symp. on Commun. Theory, pp. 317-327, 1952.

[48] C. T. Zahn and R. Z. Roskies, "Fourier descriptor for plane closed curves," IEEE Trans. Comput., Vol. c-21, No. 3, pp. 269-281, Mar. 1972.

[49] R. L. Cosgriff, "Identification of shape," Ohio State Univ. Res. Foundation, Columbus, Rep. 820-11, ASTIA AD254 792, Dec. 1960.

[50] G. H. Granlund, "Fourier preprocessing for hand print character recognition," IEEE Trans. Comput., Vol. c-21, pp. 195-201, Feb. 1972.

[51] S. R. Dubois and F. H. Glanz, "An autoregressive model approach to two-dimensional shape classification," IEEE Trans. Pattern. Anal. Machine Intell., Vol. PAMI-8, No. 1, pp. 55-66, Jan. 1986.

[52] F. Mokhtarian and A. Mackworth, "Scale-based description and recognition of planar curves and two-dimensional shapes," IEEE Trans. Pattern. Anal. Machine Intell., Vol. PAMI-8, No. 1, pp. 34-43, Jan. 1986.

[53] A. L. Yuille and T. A. Poggio, "Scaling theorem for zero crossings," IEEE Trans. Pattern. Anal. Machine Intell., Vol. PAMI-8, No. 1, pp. 15-25, Jan. 1986.

[54] C. W. Richard, Jr. and H. Hemami, "Identification of three-dimensional objects using Fourier descriptors of the boundary curve," IEEE Trans. Syst. Man Cybern., Vol. SMC-4, No. 4, pp. 371-378, July 1974.

[55] T. P. Wallace and P. A. Winz, "An efficient three-dimensional aircraft identification algorithm using normalized Fourier descriptors," Comput. Graph. Image Process., 13, pp. 99-126, 1980.

[56] E. Persoon and K. Fu, "Shape discrimination using Fourier descriptors," IEEE Trans. Syst. Man Cybern., Vol. SMC-7, No. 3, pp. 170-179, Mar. 1977.

[57] G. Lemay and J. Dessimoz, "Recovery of gray scaled images from contour processed representations," Proc. ICASSP 83, pp. 116-117, Boston 1983.

[58] G. Anderson, "Frequency-domain image representation," MIT PhD thesis, Oct. 1969.

[59] P. Margos, R. Mersereau, and R. Schafer, "Two-dimensional linear prediction analysis of arbitrarily-shaped regions," Proc. ICASSP 83, pp. 104-107, Boston 1983.

[60] S. Zucker, "Toward a model of texture," Comp. Graphics and Image Proc. 5, pp. 190-202, 1976.

[61] G. Cross and A. Jain, "Markov random field texture models," IEEE Trans. Pattern Anal. Machine Intell., Vol. PAMI-5, No. 1, pp. 25-39, Jan. 1983.

[62] F. Schmitt, "Color texture reconstruction using a bidimensional Markov model," Proc. ICASSP 82, pp. 444-447, 1982.

[63] R. Haralick, "Statistical and structural approaches to texture," Proc. of IEEE, Vol. 67, No. 5, pp. 786-804, May 1979.

[64] M. Riley, "The representation of image texture," MIT AI-TR-649, Sept. 1981.

[65] P. Chen and T. Pavlidis, "Segmentation by texture using correlation," IEEE Trans. Pattern Anal. Machine Intell., Vol. PAMI-5, No. 1, pp. 64-69, Jan. 1983.

[66] J. K. Aggaarwal and R. O. Duda, "Computer analysis of moving polygonal images," IEEE Trans. Compt, Vol. c-24, No. 10, pp. 966-976, Oct. 1975.

[67] W. B. Thompson, K. M. Mutch, and V. A. Berzins, "Dynamic occlusion analysis in optical flow field," IEEE Trans. Pattern Anal. Machine Intell., Vol. PAMI-7, No. 4, pp. 374-383, July 1985.

[68] J. L. Turney, T. N. Mudge, R. A. Volz, "Recognizing partially occluded parts," IEEE Trans. Pattern Anal. Machine Intell., Vol. PAMI-7, No. 4, pp. 410-421, July 1985.

[69] W. K. Pratt, Digital Image Processing, John Wiley & Sons, 1978.

[70] D. H. Ballard and C. M. Brown, Computer Vision, Prentice-Hall, 1982.

[71] O. R. Mitchell and T. A. Grogan, "Global and partial shape discrimination for computer vision," Optical Engineering, Vol. 23, No. 5, pp. 484-491, September/October 1984.

[72] J. W. Gorman, O. R. Mitchell, and F. P. Kuhl, "Partial Shape Recognition Using Dynamic Programming," IEEE Trans. Pattern Anal. Machine Intell., Vol. 10, No. 2, pp. 257-266, March 1988.

[73] P. E. Zwicke and I. Kiss, Jr., "A New Implementation of the Mellin Transform and its Application to Radar Classification of Ships,"

IEEE Trans. Pattern Anal. Machine Intell., Vol. PAMI-5, No. 2, pp. 191-199, March 1983.

[74] C. Myers, L. R. Rabiner, and A. E. Rosenberg, "Performance Tradeoffs in Dynamic Time Warping Algorithms for Isolated Word Recognition," IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-28, No. 6, pp. 623-635, December 1980.

[75] J. Illingworth and J. Kittler, "The Adaptive Hough Transform," IEEE Trans. Pattern Anal. Machine Intell., Vol. PAMI-9, No. 5, pp. 690-698, September 1987.

[76] D. B. Shu, C. C. Li, J. F. Macuso, and Y. N. Sun, "A Line Extraction Method for Automated SEM Inspection of VLSI Resist," IEEE Trans. Pattern Anal. Machine Intell., Vol. 10, No. 1, pp. 117-120, January 1988.

[77] F. M. Rhodes, J. J. Dituri, G. H. Chapman, B. E. Emerson, A. M. Soares, and J. I. Raffel, "A Monolithic Hough Transform Processor Based on Restructurable VLSI," IEEE Trans. Pattern Anal. Machine Intell., Vol. 10, No. 1, pp. 106-111, January 1988.

[78] K. Hanahara, T. Maruyama, and T. Uchiyama, "A Real-Time Processor for the Hough Transform," IEEE Trans. Pattern Anal. Machine Intell.,Vol. 10, No. 1, pp. 121-125, January 1988.