Feature Extraction Without Edge Detection

Ronald Dean Chaney

©Massachusetts Institute of Technology 1993
All rights reserved

Feature Extraction Without Edge Detection

by
Ronald Dean Chaney

Submitted to the Department of Electrical Engineering and Computer Science
on August 6, 1993 in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Information representation is a critical issue in machine vision. The representation strategy at the primitive stages of a vision system has enormous implications for the processing capabilities in the subsequent stages. Existing feature extraction paradigms, like edge detection, provide sparse and unreliable representations of the image information. In this thesis, we propose a novel feature extraction paradigm. The paradigm is based on the dual interpretation of the Laplacian of Gaussian (LoG) as a matched filter and an edge locator. The zero-crossings of the LoG tend to outline subjective features in the image, such as isobrightness regions. The typical size of the outlined regions depends on the spatial width of the LoG filter. Hence, a naive approach would be to take the regions bounded by the zero-crossings of the LoG filter as the features. In practice, such regions consist of multiple subjective regions that have merged together due to the smoothing process. To address this issue, we introduce a stable, robust decomposition of regions into their salient parts. The resulting subregions, called *simple region features*, serve as the feature primitives for higher level processing. The region decomposition is computed from the medial axis skeleton of each region bounded by zero-crossings. Each subregion corresponds to a portion of a branch of the medial axis skeleton; each skeleton branch is divided at positions where the distance from the skeleton to the bounding contour is minimized. To facilitate the computation of the decomposition, a number of computational geometry problems are addressed. A novel scale-space is introduced for contours and the medial axis skeleton. The scale-space is parametric with the complexity of the contour or the skeleton. The complexity measure of a skeleton is the number of branches. A related complexity measure of a contour is the number of extrema of curvature of the contour. The simple region features are dense, stable, and robust. The primary advantage of simple region features is that they have abstract geometric attributes pertaining to their size and shape. To demonstrate the utility of the feature extraction paradigm, we apply it to passive navigation. We subsequently argue that the paradigm is applicable to a variety of vision subproblems.

Thesis Supervisor: Berthold K. P. Horn
Title: Professor of Computer Science and Engineering

Acknowledgments

I wish to thank my advisor, Berthold Horn, for sharing some of his extraordinary talent and insight with me over the past few years. I also thank my thesis readers, Alan Willsky and Ellen Hildreth. Alan Willsky is enormously gifted and possesses an amazing amount of enthusiasm for his students' work. Ellen Hildreth provides extremely thoughtful, insightful commentary. I am privileged to have worked with all of them.

The MIT Artificial Intelligence Laboratory is a stimulating environment for conducting research. Director Patrick Winston deserves much credit for nurturing the lab and fostering its growth.

I must thank Eric Grimson and Berthold Horn for being extremely supportive when I was temporarily unproductive for medical reasons. The MIT Artificial Intelligence Laboratory and the MIT Department of Electrical Engineering and Computer Science displayed tremendous integrity during the same period.

I thank the Surveillance Systems Group at MIT Lincoln Laboratory. Lincoln Laboratory provided funding for the early stages of the work contained in this thesis. Gerald Morse and Leslie Novak were particularly supportive.

I thank my fellow graduate students who have made my stay at the AI Lab more enjoyable and productive. I won't list them, for fear of leaving someone out. However, Anita Flynn provides extraordinary enthusiasm and leadership and has contributed immeasurably to the lab in many ways.

Finally, I thank my family for their unending encouragement and support. I especially thank my wife, Julie, for her unshakable confidence in me.

Some of the data for this research was obtained at the Calibrated Imaging Laboratory at Carnegie Mellon University, sponsored by DARPA, NSF, and NASA.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-91-J-4038.

To my parents,
who taught me
the value of education
and the virtue of hard work

Contents

Chapter 1

Introduction

1.1 Representation in Vision Systems

The purpose of a machine vision system is to extract information about the environment from an image or sequence of images. The information obtained from the vision system must be useful for the execution of a particular task or set of tasks. The ultimate measure of success or failure of a vision system is the utility of the information that it provides.

There is an enormous quantity of information present in each image. Much of that information is pertinent for a variety of tasks. For example, biological vision systems are often cable of inferring the size, shape, and position of objects in the environment from the brightness image projected onto their retinae. Such information may be subsequently used to identify known objects and to navigate obstacles.

However, a substantial portion of the information contained in each image is irrelevant for any given task. For example, the average brightness of an image is inconsequential to most tasks. Even in the absence of clouds, the illumination of the sun varies dramatically throughout an afternoon. However, the ability of biological systems to identify objects and navigate in the environment is unaffected. In fact, biological systems are often unaware of the difference in illumination between midday and late afternoon.

The key to developing a vision system is to separate the useful information from the irrelevant data contained in the image. The goal is to transform an image consisting of an cumbersome array of brightness values into a form that makes the useful information explicit. This is a non-trivial task because it is not clear *a priori* what information is useful and how to extract it from the image.

A critical issue in the design of any vision system is the representation of information (Marr[52]). The specification of the information representation at any stage in the vision system has an enormous impact on subsequent stages of the system. The representation determines the information that is available to subsequent processes.

The choice of a representation strategy has a substantial impact on the capability of the vision system to achieve its goals.

Consider, for example, two hypothetical systems that combine depth estimates from motion and stereo algorithms. In each of the systems, depth estimates are obtained from two processing modules: depth from stereo and depth from motion. The goal of each system is to combine the depth estimates to obtain a single smooth surface representation.

In the first system, each module obtains the depth estimates and reconstructs a surface by smoothing and interpolating the estimates spatially. These two surfaces are then combined to provide the final surface. In the second system, the raw depth estimates from each module are combined prior to any smoothing and interpolation. The combined depth estimates are smoothed and interpolated to produce the final surface.

The second system has a fundamental advantage over the first. In the first system, pertinent information is discarded prior to the combination of the depth estimates. The first system is incapable of drawing a distinction between the depth values estimated directly from the data and the depth values inferred by interpolation. Due to the representation, the first system inadvertently gives equal preference to an interpolated depth estimate and a measured estimate. Ideally, the measured estimate should be given preference. In contrast, because the second system has direct access to the individual depth estimates, no such confusion exists. This advantage is a direct result of the representation chosen for each system.

This example illustrates the importance of the information representation strategy to the performance of the vision system. The two systems are tacitly assumed to possess similar information processing capabilities. However, the form of the information that is provided as an intermediate representation has an enormous impact on the overall effectiveness of the system.

1.2 Feature Extraction

One general approach for extracting information from the image is to reduce the quantity of data by making abstractions. The vision system designer makes assumptions about the form of the information present in the data. He develops the processing algorithms based on these assumptions and specifies data representations that make the pertinent information explicit.

One type of abstraction that is often used in machine vision is the definition of a feature. The definition of a feature requires the assumption that a particular occurrence in the image is significant. An implicit model specifies the relationship between the feature in the image and some event or occurrence in the environment.

For example, the most common features used by machine vision systems are edges. Edges are modeled as the location of brightness discontinuities. The brightness discontinuities are assumed to be significant because they are often associated with physical discontinuities, such as occlusion, in the environment (for example, Marr & Hildreth[53] or Canny[13]). The edges are implicitly modeled as the locations of the projections of physical discontinuities in the environment onto the image.

Manipulating the edges is considered less burdensome than manipulating the brightness values because there are fewer edges than there are brightness values. The amount of data present in the edge map is reduced compared to the original image. As a direct result, the information processing is more manageable because there is less data to process.

The selection of an abstraction or set of abstractions is critical to the performance of the overall system. The choice of an abstraction specifies the information that is available to the higher level stages of computation. If the abstraction is chosen sensibly, a higher level process may extract the desired information in a straightforward manner. However, if the abstraction is chosen poorly, a higher level process may not be able to obtain the desired information at all.

The choice of an abstraction has enormous implications for subsequent processes. For example, the edge detection process is indifferent to gradual changes in brightness over the image. In some contexts, this is beneficial. For example, gradual spatial changes in the image are often due to illumination effects; such effects are irrelevant when identifying an object. However, in other contexts, eliminating information regarding subtle brightness changes is undesirable. In such cases, edge detection is an inappropriate abstraction.

Later, we shall argue that edge detection discards too much of the image information. Furthermore, edges are unreliable and sparse. Despite their prevalence in the field, edges are not particularly good candidates for features. However, we cite them here because they are the most common information abstraction used in machine vision.

The definition of an ideal feature is somewhat elusive (for example, Richards & Jepson[64]). However, we may gain insight into the problem by considering some of the desirable properties of a feature extraction process. Ideally, the features should provide a rich description of the image and, subsequently, the environment. The feature abstraction should be chosen such that it represents the useful information explicitly and efficiently.

It is desired that each feature extracted from an image is associated with some particular object, part of an object, or some other identifiable occurrence in the environment. The set of features need not constitute a subjective description of the environment. However, each particular feature should correspond to some physical event in the environment.

The feature extraction process should provide a dense description of the image. There are occasions when portions of the image contain little or no information. Some portions of the image may consist of a nearly constant brightness value (e.g. the sky on a cloudless day). However, when meaningful brightness variations do occur in the image, it is desirable that the feature extraction process capture the information as much as possible.

It is also desirable that the feature extraction process be robust against small perturbations in the input data. From a practical point of view, the feature extraction process should be stable against small changes in illumination, viewing direction, and deformations of the objects in the environment. Otherwise, if small changes in any of these quantities lead to large changes in the features, the interpretation of such features would be difficult.

Features are desired to be stable across sequences of images when the camera moves relative to the environment. This property is actually a consequence of the properties stated above, but it is worth mentioning because of its importance. Features that persist over time facilitate the analysis of the apparent motion of the features in the image. From such analysis, we may infer, for example, the motion of the camera and the structure of the environment. The stability of the features has a direct impact on the ability of subsequent algorithms perform this analysis.

Finally, in the context of machine vision systems, it is desirable that the features be applicable to a variety of vision subproblems. If the features are used by multiple modules of the vision system, the computational burden is reduced relative to computing distinct features for each module. Furthermore, a consistent representation is beneficial for integrating information among the various processing modules.

In the next section, we argue, based on these ideas, that edge detection is not a particularly good feature extraction paradigm. Edges often correspond to the location of physical discontinuities in the environment. However, edges do not constitute a rich description of the image. Typically, edges are sparse in the image and they are sensitive to small perturbations in the image data. Furthermore, edges are typically not stable across most sequences of images. Overall, they do not provide a reliable foundation for higher level processing.

Due to the importance of the information extraction and representation at the earliest stages of the vision system, we find that it is necessary to consider alternative feature extraction paradigms. Specifically, in the following chapters, we introduce a novel feature extraction paradigm. We demonstrate that it is useful for extracting relevant information from the image by applying it to the passive navigation problem. We subsequently argue that the novel feature extraction paradigm has significant advantages over edge detection.

1.3 Critique of Edge Detection

Edge detection is the most common method for feature extraction in machine vision. The number of edge detection algorithms is enormous (for example, Roberts[67], Marr & Hildreth[53], Canny[13], and many others). There are many review articles in the literature (for example, Davis[22], Ballard & Brown[5], Rosenfeld & Kak[68], Torre & Poggio[72], and Hildreth[31],[32]). In this section, we do not provide a complete review of the subject. Rather, we consider the problem in sufficient detail to critique the general approach.

The purpose of edge detection is to convert the large array of brightness values that comprise an image into a compact, symbolic code. The goal of an edge detection algorithm is to determine the location of brightness discontinuities in the image. The brightness discontinuities are assumed to correspond to physical discontinuities in the environment like occluding contours, shadows, and changes in surface orientation.

Edge detection algorithms consist of three stages. A differentiation operation is used to locate the brightness discontinuities. A smoothing operation makes the differentiation more well conditioned and specifies the scale of the overall process. A thresholding operation chooses the candidate edges that are significant. The result is set of points that typically correspond to the brightness discontinuities in the image. In Figure 1-1, an image and its associated Canny edges[13] are depicted.

Edge detection is often viewed as a problem of numerical differentiation (for example, Bertero, Poggio, & Torre[6] and Torre & Poggio[72]). Brightness discontinuities are characterized by the maxima of some first order derivative operation, for example, the maxima of the gradient. Similarly, brightness discontinuities may be characterized by the zero-crossings of a second order derivative operation. Two prevalent second order spatial derivatives are the Laplacian[53] and the second directional derivative in the direction of the gradient[29].

Because differentiation is sensitive to noise, it is necessary to smooth the image data. A variety of optimal approaches for smoothing the image data have been proposed (for example, Shanmugam, Dickey, & Green[69], Marr & Hildreth[53], Lunscher[51], and Canny[13]). Each approach, in its own way, specifies a filter whose shape optimizes the tradeoff between the localization of the edge and the stability against noise.

Another function of the smoothing operation is to specify the resolution or scale of the edge detection process[80]. If the smoothing operation has a broad spatial support, the details in the image are removed. In this case, only the edges corresponding to the coarse structure in the image are obtained. Conversely, if a narrow support filter is used, edges pertaining to the fine detail in the image emerge.

Upon applying the smoothness and differentiation operations to the image data, the edge detection algorithm must determine which of the candidate edge locations are significant. In the case of a first order derivative operation, some of the maxima are not related to brightness discontinuities. Particularly in areas of the image that have roughly uniform brightness, maxima in the first order spatial derivative often correspond to insignificant brightness changes in the image. Consequently, edge detection algorithms apply a threshold to eliminate the insignificant edges.

The edges, like any features, are not intrinsically useful. It is the responsibility of subsequent processes to infer information pertaining to the environment from the edge map. The ultimate test for determining the utility of edge detection is the ability of subsequent processes to make such inferences. Given that making such judgements

Figure 1-1: Canny Edges. An image and its Canny edges are depicted.

is difficult or impossible, we consider edge detection based on the discussion of the previous section.

Edges often correspond to the projection of physical discontinuities in the environment onto the image. Furthermore, edge detection algorithms do succeed in providing a compact, symbolic description of the image. However, we argue that the symbolic code does not provide a rich description of the image.

In practice, edge detection provides a sparse representation of the image. The edge maps rarely characterize all of the physical discontinuities in the image. In most cases, edge detection algorithms identify only a fraction of the subjective edge locations. Gaps often exist in the edge map along the contours associated with physical discontinuities and many of the discontinuities are not represented at all. For example, in Figure 1-1, many of the boundaries between the books are not marked in the edge map, there is no information pertaining to the woman's nose, and the cup is only partially outlined.

Furthermore, edges are often reported at locations in the image where no significant physical discontinuities are present. Edges are often reported that are related to textural characteristics, rather than discontinuities. For example, many of the edges in the region associated with the floor in the Figure 1-1 are due to the texture in the floor pattern rather than significant discontinuities.

In addition, the edge detection process is sensitive to small perturbations in the image data. The thresholding mechanism that is present in all edge detection algorithms is responsible for this sensitivity. The problem arises whenever the edge discriminant (the quantity compared to the threshold) is near the threshold value. When this is the case, a small change in the brightness pattern near the candidate edge may cause a change in the decision made by the threshold mechanism.

As a result, the edges are often spurious when considered over a sequence of images. Candidate edges whose discriminant values are near the threshold often tend to appear and disappear as the sequence evolves. This type of temporal instability in the representation is undesirable. For example, such instability makes it difficult to track the edges over a sequence of images.

In summary, edge maps do not constitute a rich description of the image. Edges do not demark the image projections of physical discontinuities reliably. The edge representation is sparse and sensitive to small perturbations in the image data. These weaknesses are fundamental to edge detection because of its reliance on a threshold mechanism.

Given the shortcomings of edge detection, it is desirable to consider alternative approaches for feature extraction. A more reliable and robust method of extracting and representing information in the image is needed. An improved feature extraction paradigm would enhance the performance of higher level processes.

1.4 Thesis Overview

In this thesis, we introduce a novel feature extraction paradigm. We demonstrate the utility of the paradigm by applying it to the passive navigation problem. We argue that the paradigm is applicable to a variety of vision subproblems in addition to passive navigation. We subsequently argue that there are substantial advantages to the novel paradigm over existing paradigms, such as edge detection.

The features consist of the salient parts of regions bounded by zero-crossings of the Laplacian of Gaussian filter. The features are obtained at multiple resolutions that are determined by the spatial width of the Laplacian of Gaussian filter. We call the features *simple region features* because they consist of subregions of the zero-crossing regions that have simple shapes.

In Part I of the thesis, we develop a number of capabilities in the realm of computational geometry. The capabilities are necessary for the computation of the simple region features. Previous algorithms are not sufficient to compute the smooth contours and the corresponding medial axis skeletons that are required by the simple region feature extraction process.

We treat the results pertaining to computational geometry independently because the results have implications beyond the simple region feature extraction paradigm. The capabilities we introduce are useful for representing, interpreting and ultimately recognizing contours and regions bounded by contours. We consider these possibilities briefly in Part I before moving on to the definition of the simple region features.

In Chapter 2, we define an analytical contour representation. The representation has the advantage that a contour is represented as a mathematical curve rather than a list of points. The curve is continuous, and a number of properties, such as the position, orientation, and curvature are represented explicitly.

In Chapter 3, we consider the computation of the medial axis skeleton from the contour representation. The skeleton is also represented analytically. In contrast to most existing algorithms, the skeleton computation is robust against small perturbations in the data points of the bounding contour.

The analytical contour and skeleton representations facilitate the computation of a novel scale-space for contours and the medial axis skeleton. The scale-space achieves an explicit tradeoff between the complexity and the accuracy of the representation. The complexity is alternately measured as the number of extrema of curvature present in the contour, or the number of branches of the skeleton. The accuracy is measured by the square-error between the bounding contour and the data points.

In Part II we define a novel feature extraction paradigm. We demonstrate the viability of the paradigm by applying it to the passive navigation problem. The approach achieves reasonable results for determining motion and structure from motion. We subsequently argue that the paradigm has broader implications than passive navigation; simple regions are the foundation of a novel early vision paradigm.

In Chapter 5, we define the *simple region feature extraction paradigm.* Simple region features consist of the salient parts of regions bounded by zero-crossings of the Laplacian of Gaussian filter. The zero-crossings are smoothed using the techniques described in Chapter 2. The medial axis skeleton is computed as described in Chapter 3. A decomposition is defined that breaks the regions bounded by zero-crossings into salient parts. These subregions serve as the features.

In Chapter 6, we develop an algorithm for tracking the features across multiple frames of an image sequence. We consider a recursive algorithm for estimating the optical flow or two-dimensional image velocity of each feature. In addition to the optical flow, the estimator provides a measure of the reliability of each estimate.

In Chapter 7, we use the optical flow to determine the direction of translation of the camera and estimate the relative depth of each feature. We use the reliability measure to weight the flow estimates when determining the direction of translation of the camera. The reliability measure also provides a measure of the reliability of the depth estimates.

In Chapter 8, we discuss the merits of the simple region feature extraction paradigm and consider future directions of the research. The primary advantage of the simple region feature extraction paradigm over existing paradigms is that simple regions have spatial extent; that is, they have shape and size attributes. Throughout the development and demonstration of the approach we exploit these attributes whenever possible.

We argue that the simple region feature extraction paradigm has broader application than passive navigation. We briefly consider two additional problems, stereopsis and object recognition, that may be addressed using simple region features. We argue that by using the same feature abstraction for a variety of problems, integration of information from the individual modules becomes more straightforward. In this context, simple region features are the foundation of a novel early vision paradigm.

Finally, in Chapter 9, we draw some conclusions.

Part I

Computational Geometry

Chapter 2

Analytical Contour Representation

2.1 Introduction

The interpretation and recognition of noisy contours, such as silhouettes, have proven to be difficult. One obstacle to the solution of these problems has been the lack of a robust representation for contours. Improvements in the representation of contours and the ability to manipulate the representation will lead to improvement of interpretation and recognition procedures. Furthermore, because manipulating contours is useful as a primitive operation in other contexts, a more robust representation for curves is likely to lead to improved performance in a variety of higher level functions.

Curvature has long been recognized as an important property of contours. In 1954, Attneave[4] published an important paper in which he observed that extrema of curvature along contours provided much of the information necessary to recognize objects from line drawings. Attneave manually picked points that corresponded to extrema of curvature and connected the points with straight lines. Remarkably, recognition based on such figures was found to be simple for human observers. Thus, curvature is likely to play an important role in the development of computer vision recognition systems.

Consequently, many procedures for interpretation and recognition of contours require estimates of curvature (see Section 2.2). Unfortunately, curvature is a second order derivative property of the coordinates of the curve. Because contours extracted from real data are almost always noisy, curvature estimates are typically unreliable. Therefore, it is critical that a robust, reliable method for smoothing contours be available. Furthermore, it is essential that the representation provide a robust estimate for curvature.

One barrier to the development of a robust contour representation paradigm is that geometric relationships are very awkward to represent computationally. In particular, the discrete nature of computers and the unavoidable roundoff error make inference of simple geometric properties extremely difficult[78]. Consider a trivial example. Two points in the continuous domain, (x_1, y_1) and (x_2, y_2) determine a line and the midpoint, $(\frac{x_1+x_2}{2}, \frac{y_1+y_2}{2})$, is guaranteed to be on the line. However, in the discrete case any computation of the distance between the midpoint and the line almost always yields a non-zero result. This leads to the paradoxical conclusion that a point on the line has a non-zero distance from the line.

In this chapter, we present a novel approach to representing contours. The curve is represented by a list of pairwise tangent circular arcs. To account for roundoff error, the representation paradigm uses a definition of tangency that is different from the pure mathematical definition. A variety of mathematical properties, including curvature and orientation are represented explicitly. A variety of global properties of the curve are computed easily from the representation.

We also introduce a novel approach to contour smoothing. We optimize a tradeoff between the complexity of the curve and the proximity of the curve to the data points. The complexity of the contour is measured by the number of extrema of curvature of the curve. Once a particular complexity has been specified, the curve that minimizes the square-error between the data points and the curve is chosen. A multiple scale description of the curve is obtained by computing curves with a variety of complexities.

In Section 2.2, we consider previous attempts to represent and smooth contours. In Section 2.3, we consider the definition of the contour representation and the mechanics of deforming the contour. In Section 2.4, we introduce the novel smoothing criteria and complexity scale-space. In Section 2.5, we consider the computation of a number of useful mathematical properties from the representation. In Section 2.6, we summarize.

2.2 Background

A variety of methods of representing and smoothing contours have been proposed. Of course, the representation of the contour has a tremendous impact on the ability to interpret or recognize an object bounded by the contour. In this section we consider a sample of existing contour representation paradigms. In particular, we focus on the implications of the representation on the ability to recognize and interpret contours.

Perhaps the simplest representation for a contour is a list of the coordinates of points along the curve (for example, Hoffman & Richards[36], Mokhtarian & Mackworth[57] and Lowe[49]). A curve represented in this fashion may be smoothed by applying a Gaussian filter to the x and y coordinates, independently. Estimates of the orientation and curvature of the contour may be obtained from finite difference approximations of the derivatives of the local coordinates.

Unfortunately, Gaussian filtering applied to coordinates of points suffers from a well-known shrinkage problem[42]. As the standard deviation of the Gaussian filter increases, the perimeter of the smoothed curve is guaranteed to decrease. In fact, as the standard deviation tends to infinity, the coordinates of the smoothed curve converge to a single point. Therefore, the smoothed curve is guaranteed to stray from the original data points; there is a bias toward the interior of the contour.

Lowe[49] proposes a smoothing procedure that compensates for the shrinkage problem explicitly. This procedure reduces, but does not completely eliminate, the shrinkage problem. Lowe points out that locally, the shrinkage problem manifests itself as a tendency of the smoothed points to migrate toward the center of curvature. Based upon the amount of smoothing and an estimate of the curvature, it is possible to predict the amount of shrinkage that would occur with straightforward filtering. Lowe's algorithm explicitly compensates for the shrinkage based on this estimate.

Horn and Weldon[42] propose a solution to the shrinkage problem. They represent the curve by its extended circular image[40]. The extended circular image is obtained by mapping the curvature of a particular point on the curve to the location of the circle that has the same orientation of the point of interest. Thus, the extended circular image is the curvature of the curve as a function of the orientation of the curve. (See Section 2.5.5.) Horn and Weldon show that applying a filter with unit DC gain to the extended circular image leaves the perimeter of the underlying curve unchanged. Therefore, their method does not suffer from the shrinkage problem. Filtering the extended circular image is simply smoothing the curve in a different domain. Unfortunately, the procedure is only applicable to convex curves. Attempts to extend the procedure to general curves have proven unsuccessful.

An alternate method is to represent the contour by a set of line segments. The line segments are chosen using a split and merge algorithm (see Horn[38], Horowitz & Pavlidis[62], [43], Chen & Pavlidis[18], and Grimson[27] pp. 104-105). The representation is constructed recursively. The algorithm begins with a single line segment that is defined by the two endpoints of the curve. The data point that deviates the farthest from the line segment is found. If the distance of this point from the original segment is greater than a specified tolerance parameter, the segment is split into two. Each new segment is defined by one of the original endpoints and the point of maximum deviation. This process is repeated, recursively, until all data points are within the specified distance of the curve. Neighboring segments are combined into a single segment if doing so would not cause the distance from any data point to the segment to exceed the tolerance.

The line segment representation based upon the split and merge algorithm provides a simple, computationally efficient means of representing the contour. The amount of smoothing is controlled by the tolerance parameter. Orientation estimates are obtained from the orientation of each line segment. Curvature estimates may be obtained from the change in orientation of neighboring line segments and the length of neighboring line segments. Unfortunately, the curvature estimates obtained from this representation are only marginally useful.

It is also possible to represent a contour with circular arcs and line segments. A popular method of acquiring this representation is to map the curve into Θ-s space.

At each point along the curve, estimates of the orientation and the arclength between the point and the previous point are obtained. An estimate of the orientation of the curve as a function of arclength is obtained. In this space, called Θ-s space, a horizontal line corresponds to a line segment on the curve. A non-horizontal line corresponds to a circular arc; the curvature of the arc is equal to the slope of the line. By making a piecewise linear least square-error fit to the data in Θ-s space, a representation of the curve by a set of circular arcs is implicitly obtained. This procedure is described in more detail in Grimson[27] pp. 105-108. The use of Θ-s space is particularly popular for object recognition systems (for example, McKee and Aggarwal[55], Perkins[63], Turney *et al*[75], Clemens[20], and Grimson[27], [28]).

Estimates of orientation and curvature may be obtained directly from the circular arcs and line segments in the representation. The piecewise linear fit in Θ-s space is an explicit method of smoothing the curve. However, strictly speaking, the orientation is discontinuous because neighboring arcs and segments are not guaranteed to be tangent. Thus, the curvature is infinite at these points. This problem may be alleviated by a variety of *ad hoc* methods; however, such methods necessarily lead to curvature estimates that are inconsistent with the representation.

Curvature estimates play a major role in most contour interpretation and recognition algorithms. In many cases, such as codon coding[36], the curvature estimates play an explicit role. In other cases, such as the medial axis transform[8], the curvature of the contour has an implicit, but important, effect on the calculation.

The medial axis transform is an algorithm that obtains a graph with the same topology as the region it represents. Each point on a branch of the graph is equidistant from two points on the bounding contour of the region. A node of the graph is a point equidistant from three or more points on the contour. The graph is often called the medial axis skeleton or skeleton for short. The skeleton is useful because it decomposes the region into simpler parts. Each part is represented as a branch of the skeleton. Furthermore the skeleton provides a convenient representation of the topological relationships of the parts of the region.

We may consider the skeleton to include information about the distance between each point on the graph and the bounding contour. That is, for each point on the branch of a graph we assume the distance between that point and the two closest points on the contour are known. In this case, there is a unique mapping from the skeleton to the bounding contour of the region and vice versa.

Given the unique mapping between the skeleton and the bounding contour of the region, it is not surprising that the skeleton is highly dependent on the curvature of the contour. Each branch of the skeleton that terminates into the contour (rather than into a node of the skeleton) does so at a positive maximum of curvature. Furthermore, there is a simple test to determine if a positive maximum of curvature corresponds to a terminus of a branch of the contour. If the osculating circle at the maximum of curvature lies in the interior of the contour, the maximum is associated with a terminus of a branch of the skeleton. Otherwise, there is no terminus associated with the maximum. Consequently, the local curvature of the bounding contour plays an important role in the topology of the skeleton. In fact, the well-known sensitivity of the skeleton to small perturbations in the contour is completely characterized by the effect of the perturbation on the local curvature.

Therefore, the effect of the contour representation on curvature is critical to the computation of the medial axis skeleton. Typically, the sensitivity of the skeleton to perturbations in the contour is reduced by computing an approximation to the medial axis skeleton. For example, Leymarie & Levine[47] compute a skeleton that minimizes a cost function that includes the distance from the true medial axis skeleton and a smoothness term for the resulting skeleton. Such an approach may provide a reasonable result for the skeleton itself. However, the resulting skeleton and the bounding contour are inconsistent. Such inconsistency is undesirable.

Codon coding, popularized by Hoffman & Richards[36], is another method for interpreting the region bounded by a contour that makes use of curvature estimates explicitly. A codon is a portion of the curve delimited by two minima of extrema. It is desirable to decompose the curve in this fashion because the boundary of subjective parts of an object often occur at negative extrema of curvature. Thus, the list of codons provides an explicit interpretation of the contour as the set of its salient parts.

The work of Richards *et al* [36], [37], [65] on codons has been widely referenced in the literature. However, as a practical matter, the direct implementation of these ideas has been elusive. The largest obstacle has been the inability of contour representation and smoothing schemes provide reliable estimates for curvature.

Asada and Brady[3] construct a "primal sketch" for contours. A set of primitives are delimited by positions of "significant curvature changes." The primitives are detected and localized across a variety of resolutions of Gaussian filtering. The primitives include corners and "smooth joins" (large discontinuities in curvature) as well as compound primitives such as ends (two nearby corners of the same sign), cranks (nearby corners of opposite sign), bumps and dents (two nearby cranks). Presumably, such an intermediate representation could be used by a higher level process to make inferences about the contour.

Mokhtarian and Mackworth[57] propose an alternative method of describing the shape of a contour. The curve is represented by a list of coordinates along the curve; smoothing is accomplished with a Gaussian filter. The shape is represented by the position of the zero-crossings of curvature as a function of arclength along the curve. A scale-space representation reminiscent of Witkin[80] is obtained by considering the position of the zero-crossings parametric in the amount of smoothing applied to the data points. Mokhtarian and Mackworth develop a contour recognition algorithm that compares the scale-space representation of a contour to the elements of a library of such contours.

We have considered only a limited sample of existing methods for representing contours. We have touched on some of the issues related to smoothing, interpretation, and recognition of the contours. In particular, we have noted that curvature and orientation estimates often play critical roles in interpretation and recognition systems. An historical review of contour representation prior to 1980 may be found in Pavlidis[61]. For a mathematical treatment of curves and their properties, see Koenderink[45].

2.3 Computation of the Contour Representation

Any sufficiently well-behaved curve may be approximated by a set of pairwise tangent circular arcs. Such a representation is desirable because it provides a richer, more meaningful description of the contour than do traditional representation schemes. The representation provides several mathematical properties explicitly and facilitates the analytical computation of a variety of others.

The computation of the pairwise tangent arc representation is nontrivial. For example, simple geometric properties, such as tangency, are not conveniently computed by digital processors having finite accuracy. In this section, we consider the computation of the contour representation from a set of sample points along the contour.

2.3.1 Definitions

We must define terms necessary for describing the contour representation. While these terms may have widely accepted geometric meanings, it is necessary to be precise when describing their meaning in a computational context. Due to the quantized nature of computers, it is necessary to define suitable approximations to terms such as tangency.

A simple closed curve divides the plane into two simply connected regions: the interior and the exterior. An observer traversing the curve with the interior to his left is said to be moving in the positive direction of the curve. If the curve turns to the left, the curvature is said to be positive; alternatively, if the curve turns to the right, the curvature is said to be negative.

For any point along the curve, the normal vector is defined as the vector perpendicular to the curve pointing in the direction of the exterior. The normal vector specifies the orientation of the curve at the point. The angle of the normal vector is called the angle of orientation. The angle of orientation increases when traversing a segment of positive curvature. Conversely, the angle of orientation decreases when traversing a segment of negative curvature.

A circle is a locus of points equidistant from a particular point (x_c, y_c), the center of the circle. The distance from the center to any point on the circle is the radius, R. An arc of a circle is delimited by two end angles denoted by θ_1 and θ_2. The curvature of an arc is denoted by κ, and

$$|\kappa| = \frac{1}{R}. \tag{2.1}$$

The curvature is positive if the arc is traversed in the counterclockwise direction around the center, negative if the arc is traversed in the clockwise direction. The coordinates of the arc parametric in arclength may be expressed as

$$x(s) = x_c + R\cos(s\kappa + \theta_1), \tag{2.2}$$
$$y(s) = y_c + R\sin(s\kappa + \theta_1), \tag{2.3}$$

where s is the distance traversed along the arc.

Two circles are said to be externally tangent iff the distance between their centers is equal to the sum of their radii. Because it is impossible to compute distances exactly, we must use an approximation to this definition for computational purposes. Thus, two circles are considered to be externally tangent iff the difference of the sum of their radii and the distance between their centers is less than some parameter, ϵ. Specifically, two circles are externally tangent iff

$$\left|R_1 + R_2 - \sqrt{(x_{c1} - x_{c2})^2 + (y_{c1} - y_{c2})^2}\right| < \epsilon. \tag{2.4}$$

Two circles are said to be internally tangent iff the sum of one radius and the distance between the centers is equal to the other radius. Again, we must allow for quantization. Without loss of generality we assume $R_1 < R_2$. Under this condition, the circles are internally tangent iff

$$\left|R_1 - R_2 + \sqrt{(x_{c1} - x_{c2})^2 + (y_{c1} - y_{c2})^2}\right| < \epsilon. \tag{2.5}$$

Note that the parameter, ϵ, is dependent on the precision of computation. As the precision of the computation device increases, the necessary value of ϵ decreases. Thus, ϵ is not an internal parameter that significantly affects the outcome of the computation. It is a computational necessity due to quantization error.

Two circles intersect when the distance between their centers is less than the sum of their radii but greater than the magnitude of the difference of their radii. For consistency, we must add the condition that the circles are not tangent as defined above. So two circles intersect iff they are not tangent as defined above and

$$|R_1 - R_2| < \sqrt{(x_{c1} - x_{c2})^2 + (y_{c1} - y_{c2})^2} < R_1 + R_2. \tag{2.6}$$

Two circles are said to be external when the distance between their centers is greater than the sum of their radii. Two circles are external iff they are not tangent and

$$R_1 + R_2 < \sqrt{(x_{c1} - x_{c2})^2 + (y_{c1} - y_{c2})^2}. \tag{2.7}$$

Circle1 is said to be internal to circle2 when circle1 lies completely in the interior of circle2. Circle1 is internal to circle2 iff the circles are not tangent and

$$R_2 - R_1 > \sqrt{(x_{c1} - x_{c2})^2 + (y_{c1} - y_{c2})^2}. \tag{2.8}$$

A curve is represented by an ordered list of circular arcs. Each arc is specified by its center point and curvature. The end angles of each arc are determined by the points of tangency with the neighboring arcs. Neighboring arcs that have the same sign of curvature must be internally tangent; neighboring arcs that have differing signs of curvature must be externally tangent.

2.3.2 Curve Initialization

In this section we consider the computation of an arbitrary curve that passes through each data point. Once such a curve is found, it may be deformed to find a suitably smooth curve. We consider the computation of deformations of a curve and a particular smoothness criterion below.

At each data point, a circular arc is computed that passes through the point. Since three points determine a circle, two other points must be specified. The midpoint between the point of interest and one of its neighbors is used as the second point. The midpoint between the point and its other neighbor is the third. The circle determined by these points is used as the initial arc for the point of interest.

The initial arcs from two neighboring points are guaranteed to intersect at the midpoint between the points. A third arc must be computed that is mutually tangent to each initial arc. The radius of the third arc is arbitrarily chosen and its position is defined by tangency constraints. Figure 2-1 illustrates the computation of the initial curve.

This initialization procedure yields an arbitrary curve that passes through each data point. The initial curve must be deformed to obtain a more reasonable representation of the data. The computational means of the curve deformation are described in the next section. Later, we introduce a reasonable criterion for choosing an appropriately smooth curve.

2.3.3 Deformation of the Curve

In this section we consider the mechanisms for deforming a curve locally. There are three types of deformation of interest to us. First, we consider changing the curvature of a single arc. Next, we consider the rotation of two neighboring arcs without modifying their curvature. Finally, we consider the deformation of a single arc into two separate arcs. These operations provide the ability to transform the curve into a more desirable curve. In Section 2.4, we shall consider the criteria for choosing deformations of the curve such that they lead to a more desirable curve.

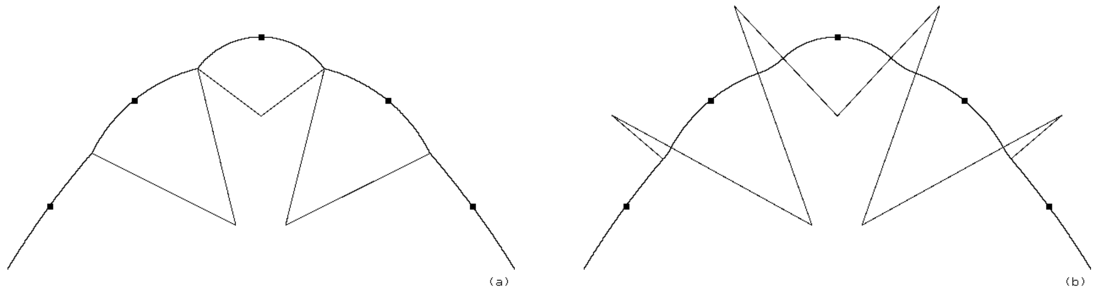


Figure 2-1: Initialization of a curve from the data points. The small squares indicate the location of the data points. The lines protruding from the curve represent radii of the circular arcs. Each arc in (a) passes through a data point and the midpoints between the data point and its neighbors. Radii are drawn from the center of curvature of each arc to the corresponding midpoints between the data points. In (b) additional arcs have been added between each neighboring pair so that each arc is tangent to its neighbor. Radii are draw from the center of curvature to the endpoints of each arc.

Modification of the Curvature of a Single Arc

Consider the deformation of a curve by modifying the curvature of a single arc. We shall call the arc of interest the new arc and the adjacent arcs neighbor1 and neighbor2, respectively. During the operation the circles associated with the neighboring arcs are considered to be fixed.

A number of constraints must be maintained during the deformation. First, the new arc must be tangent to each neighbor. Furthermore, the type of tangency must be consistent with the sign of curvature. For example, if neighbor1 and the new arc have the same sign of curvature, they must be internally tangent. Conversely, if they have opposite signs of curvature they must be externally tangent. The same constraints apply to the new arc and neighbor2.

Because the new arc must be tangent to each of its neighbors, the center point is constrained to lie on a locus of points specified implicitly by

$$\left|\sqrt{(x_{cn}-x_{c1})^2+(y_{cn}-y_{c1})^2}\pm R_1\right| = \left|\sqrt{(x_{cn}-x_{c2})^2+(y_{cn}-y_{c2})^2}\pm R_2\right|, \quad (2.9)$$

where $\{x_{c1}, y_{c1}, R_1\}$ and $\{x_{c2}, y_{c2}, R_2\}$ denote the respective neighboring circles and (x_{cn}, y_{cn}) denotes a legal center of curvature for the new arc. The value of the left side of the equation is the distance from the new center point to the point of tangency with neighbor1 and the right side is the same measure for neighbor2. The value of either side of the equation is the radius of the new circle.

The curves defined by Equation 2.9 are ellipses and/or hyperbolae. An ellipse is the locus of points whose distances from two fixed points (called the foci) have a constant sum (see, for example Thomas and Finney[71], p.411). The foci are the two centers of the neighboring arcs. The constant sum is equal to $|R_1 \pm R_2|$, where the plus or minus sign depends on the relationship of the two circles.

Similarly, a hyperbola is the locus of points whose distances from two fixed points (called the foci) have a constant difference (see, for example Thomas and Finney[71], p.418). The foci of the ellipse are the centers of the neighboring arcs. The constant difference is equal to $|R_1 \pm R_2|$ where the plus or minus sign depends on the relation of the two neighboring circles.

Equation 2.9 may specify as many as four distinct curves on the plane. However, two of the curves are eliminated by the constraint that the new arc have the appropriate tangency type with its neighbors. The two remaining curves correspond to the cases where the new arc has positive and negative curvature, respectively.

However, computing points that satisfy equation 2.9 is non-trivial. Rather than solving the equation, an iterative procedure is employed to compute legal center points for the new arc. Upon computation of the center point, the radius and, therefore, the curvature of the new arc are determined by computing the distance from the new center to either of the neighbors (using either side of equation 2.9).

For each legal center point, there is a unique point of tangency with neighbor1 and also with neighbor2. Conversely, each point on the circle of neighbor1 is associated with a particular legal center point. That is, the choice of a tangent point with neighbor1 uniquely specifies the center point and therefore uniquely specifies the tangent point of the new arc and neighbor2.

Therefore, by sweeping the location of the tangent point along the circle of neighbor1, the algorithm may implicitly sweep along the locus of legal center points. In essence, there is a single degree of freedom for choosing the new arc. The point of tangency of the new arc with the circle of neighbor1 is a computationally convenient parameterization for the operation of deforming the curvature of an arc.

Given a point of tangency on the circle of neighbor1 it is necessary to compute a center point and radius that specify a circle tangent to both neighbors. It is instructive to note that when two circles are tangent, the centers and the tangent point are collinear. Consequently, the center of the new arc must lie on the line determined by the center of neighbor1 and the point of tangency. It is necessary to search along this line systematically to determine the appropriate center point for the new arc. The modification of the curvature of a single arc is illustrated in Figure 2-2.

There is a particular choice of the center point that causes the arclength of one of the neighbors to be zero. When this occurs, the zero-length neighbor is eliminated from the representation. This case also acts as a delimiter since deforming the curvature of the new arc further would lead to an illegal configuration of the arcs.

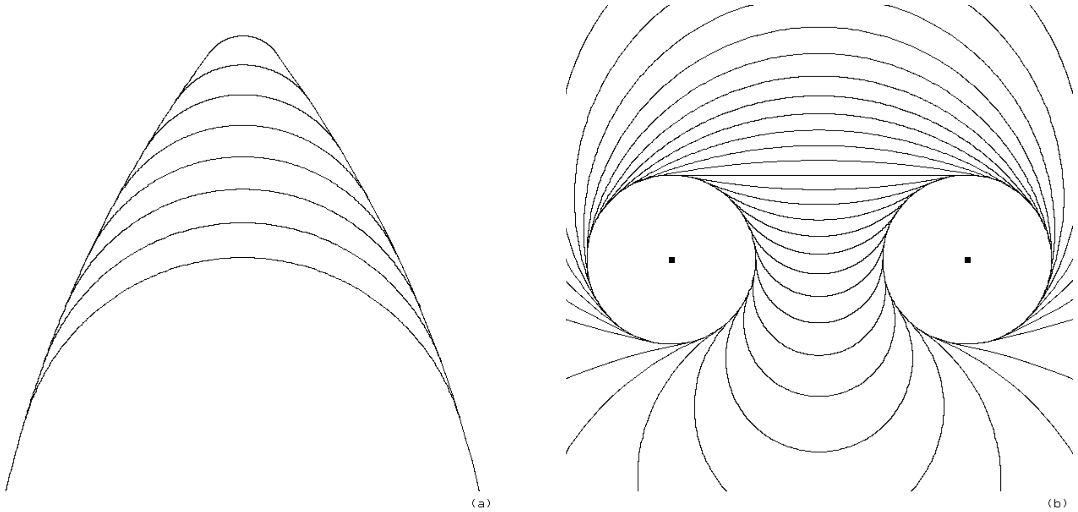


Figure 2-2: Deformation of the curve by changing the curvature of a single arc. In (a) the curves represent candidate arcs that are internally tangent to two neighboring arcs. In (b) the curves represent candidate arcs that are externally tangent to two circles.

Rotation of Two Neighboring Arcs

Consider the deformation of a curve by rotating two neighboring arcs. In this case, the two arcs change position rather than curvature. We refer to the two arcs as arc1 and arc2. We refer to their neighbors as neighbor1 and neighbor2, respectively. The circles of neighbor1 and neighbor2 are considered fixed during the operation.

First, consider circle1 with a fixed center and radius. Consider circle2 with a fixed radius but a variable center point. Circle2 is allowed to move under the constraint that the two circles remain tangent. Under these conditions, the center of circle2 follows the path of a circle whose center is coincident with the center of circle1. Moving circle2 in this manner is, therefore, equivalent to rotating circle2 about a center of rotation coincident with the center of circle1.

Now, if the position of arc1 is modified such that it remains tangent to neighbor1, it may be described by a rotation about the center of neighbor1. After the modification, arc1 and arc2 are no longer tangent. It is necessary to compute the appropriate position for arc2 such that it is tangent to arc1 and neighbor2. Since arc2 must remain tangent to neighbor2, this modification of position may be described as rotation about the center of neighbor2.

The rotation of two neighboring arcs is an operation with a single degree of freedom. If the position of arc1 is perturbed by a small amount then the position of arc2 must be changed also to maintain tangency between arc1 and arc2. A computationally convenient parameterization of this operation is the tangent point of arc1 and neighbor1. This tangent point (along with the constraint that tangency type must

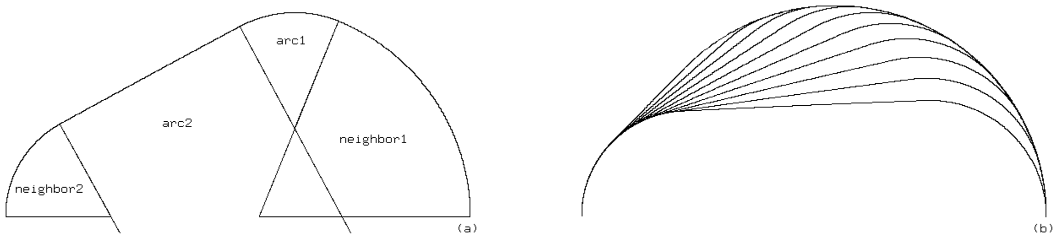


Figure 2-3: Deformation of the curve by rotating two neighboring arcs. An initial arrangement of two arcs and their neighbors is shown in (a). Candidate deformations of the curve involving a change in position of arc1 and arc2 are shown in (b). The curvatures associated with arc1 and arc2 are not affected by the operation.

be consistent with the curvature) uniquely specifies the position of arc1. Given the positions of the arc1 and neighbor2, there are typically two positions of arc2 for which arc2 is mutually tangent to arc1 and neighbor2. It is always possible to determine which of these two positions is appropriate for arc2 based on the original position of arc1 and arc2. Typically, this ambiguity is of little consequence in practice.

Once an appropriate tangent point between arc1 and neighbor1 is chosen, it is necessary to compute the new center points of arc1 and arc2. The center point of arc1 is determined by computing the appropriate point on the line specified by the center of neighbor1 and the desired tangent point. The new center point of arc2 is determined by searching for the tangent point between arc2 and neighbor2 that results in the appropriate tangency between arc2 and arc1. This operation is illustrated in Figure 2-3.

There is a particular choice of rotation for which one of arc1, arc2, neighbor1, or neighbor2 has exactly zero arclength. When this rotation is chosen, the zero-length arc is eliminated from the representation. This case also acts as a delimiter since further rotation would lead to an illegal configuration of the arcs.

Splitting an Arc into Two Arcs

Consider the deformation of a single arc into two separate arcs. This is accomplished by choosing two arcs such that they are tangent to their respective neighbors and to each other. This operation provides a mechanism for increasing the number of arcs in the curve representation. We refer to the new arcs computed in this operation as arc1 and arc2 and their neighbors as neighbor1 and neighbor2.

This operation, as defined in the previous paragraph, has three degrees of freedom. Consequently, it is necessary to provide an additional constraint. This is accomplished by constraining the tangent point between arc1 and arc2 to lie on a line that is tangent to the original arc. The choice of the constraint line is dependent on the context of

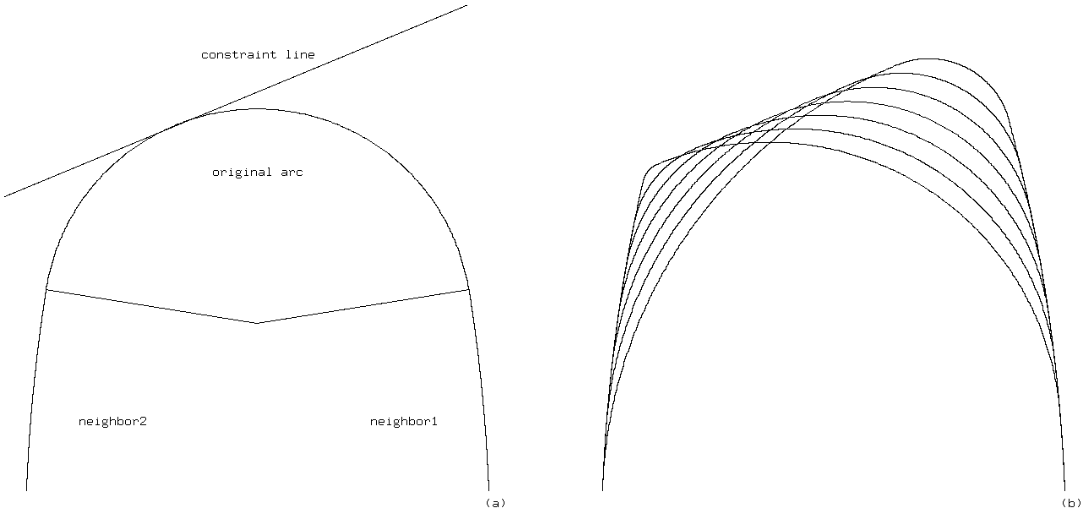


Figure 2-4: Deformation of the curve by splitting a single arc into two arcs. In (a) the original configuration of an arc and its neighbors is shown. In (b) each member of the family of curves consists of the two original neighbors joined by two new arcs. The point of tangency between the new arcs lies along the constraint line illustrated in (a).

the computation. Criteria for the choice are described Section 2.4.3. The constraint eliminates two degrees of freedom leaving only one.

Again, a convenient parameterization for this operation is the tangent point between neighbor1 and arc1. For a particular choice of the tangent point, a circle is computed that is tangent to neighbor1 and the constraint line. The point of tangency between arc1 and the constraint line is taken to be the point of tangency between arc1 and arc2. Arc2 is determined by computing the circle tangent to arc1 and neighbor2 with the specified point of tangency between arc1 and arc2. This operation is illustrated in Figure 2-4.

Once again, there is a particular deformation that leads to an arclength of zero for one of the neighboring arcs. If this deformation is chosen, the arc with zero arclength is eliminated. As above, this particular choice of parameters is a limiting case for legal deformations of this type.

2.4 Contour Smoothing

A smoothing operation on a contour involves a tradeoff between some measure of smoothness and the proximity of the curve to data points. Often, smoothness of a contour is measured by some function of the magnitude of the curvature[39]. However, in this chapter, we consider a novel approach to the tradeoff. We propose that the complexity of the curve, rather than the magnitude of the curvature, be minimized.

2.4.1 Smoothness Criterion

We propose a two-fold criterion for smoothness and proximity. The first part of the criterion is that the contour shall have minimum complexity as measured by number of curvature extrema, as described below. The second part of the criterion requires that for a given complexity, the contour shall be chosen to minimize the square-error between the contour and the data points.

A reasonable measure of the complexity of a curve is the number of extrema of curvature of the curve, M. As described in Section 2.2 Hoffman and Richards[36] have used extrema of curvature for interpreting and classifying contours. Their approach, called "codon coding", is based on the observation that extrema of curvature occur at the natural break points of the contour. That is, a human observer asked to break a contour into salient parts would place the breaks at extrema of curvature. Thus, the number of extrema of curvature of the contour is related to the number of subjective "parts" of the contour. Hence, reducing the number of extrema of the contour is tantamount to reducing the number of features that may be encoded in the contour.

Of course, there must be a reasonable criterion for deciding how many extrema and, therefore, how many features of the contour to retain in the representation. A fundamental tradeoff exists between the number of extrema of curvature and the proximity of the curve to the data points. A reasonable method of quantizing this tradeoff is to constrain the curve to pass within a specified tolerance, δ, of each data point. When the tolerance is small, it is necessary that the curve have a relatively large number of extrema to meet the constraint. When the constraint is relaxed, fewer extrema are required.

The proximity constraint defines a *tolerance circle* about each data point. The center of each tolerance circle is the associated data point and the radius is the tolerance, δ. The curve is constrained to pass within each tolerance circle. Thus, we seek to find an instance of the curve that has the minimum complexity measure and passes within each tolerance circle.

In general, there are infinitely many curves that meet the proximity constraint and minimize the complexity. From these, it is desirable to choose the one that minimizes the square-error between the data and the curve. The result is a curve that is smooth in the sense has the minimum complexity and close to the data in the square-error sense.

The tolerance acts as a scale parameter. As the tolerance is increased, fewer extrema and, therefore, fewer subjective features are present in the contour representation. Conversely, for smaller values of the tolerance, more subjective features are present. Hence, the representations computed for differing values of δ lead to descriptions of the contour with differing level of detail.

The two-fold criterion leads us to a two-stage algorithm. The first stage of the smoothing algorithm computes a curve that has the minimum complexity, M, under the constraint that the curve must pass within each tolerance circle. The second stage of the smoothing algorithm seeks the curve that has complexity measure M and minimizes the square-error between the data and the curve.

2.4.2 Computation of the Minimum Complexity Curve

An iterative procedure is used to compute the curve of minimum complexity given a particular tolerance, δ. At each iteration the algorithm seeks to reduce the difference of curvature between neighboring extrema of curvature. In doing so, the number of extrema are decreased when the curve is deformed such that the difference in curvature of a maximum and a neighboring minimum becomes zero. At each step, the curve must remain within the tolerance circle for each data point. Deformations that would move the curve outside any tolerance circle are disallowed.

Consider a case where a maximum of curvature exists during an intermediate stage of the computation. The algorithm attempts to decrease the curvature of the arc by deforming it as described in Section 2.3.3. If the algorithm finds an arc that has reduced curvature and passes within the tolerance of all the relevant points, the new arc replaces the original arc; the endpoints of the neighboring arcs are updated appropriately. Conversely, in the case of a minimum of curvature, the algorithm attempts to increase the curvature of the extremum arc. Again, if the algorithm finds an arc that increases the curvature and passes within the tolerance of the relevant data points, the arc is replaced.

Each of these operations always leads to a decrease in the arc lengths of the neighboring arcs. Often the neighboring arcs are "engulfed" in the process. That is, when the arclength of one of the neighbors becomes zero, the arc is removed from the representation. During the process, the number of extrema of curvature is reduced as multiple arcs are regrouped into single arcs.

Each extremum of curvature is updated iteratively until it is no longer possible to find an improvement. At this point, each arc in the representation that is an extremum of curvature is tangent to a "tolerance circle" around one of the data points. We call such a data point a *critical point*. If an extremum arc were not tangent to the tolerance circle of a critical point, it would be possible to modify the curvature further. The local deformation of a curve to find the minimum number of extrema of curvature is illustrated in Figure 2-5.

At this point, there is no guarantee that the curve has the minimum possible number of extrema of curvature. Consequently, a verification algorithm is applied to the curve. The purpose of the verification algorithm is to determine if each extremum is necessary given the constraints of the tolerance circles. If each extremum is verified as necessary, an instance of the minimum complexity curve has been found and the first stage is complete. However, if there exist extrema that are not verified, the verification algorithm yields information that is helpful for choosing the appropriate deformation to reduce the number of extrema of curvature. Extrema that may be eliminated by an appropriate deformation of the curve are called *nonessential extrema*.

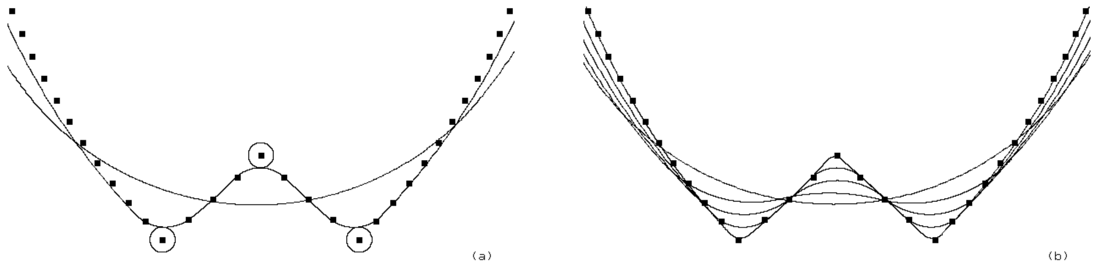


Figure 2-5: The reduction of the complexity of the curve. The two curves in (a) differ in their complexity measure by two. The curve that more closely follows the data points has three extrema of curvature. The extrema are indicated by the tolerance circles drawn about the critical data points. Note that the curve with the smaller complexity measure must deviate more severely from the data points. These curves result from different values of the scale (tolerance) parameter. The curves in (b) illustrate a possible set of intermediate states of the curve as it is transformed from higher to lower complexity. At each stage of computation, the curve is deformed such that the curvature at each maximum is reduced and the curvature at each minimum is increased.

The verification algorithm is based on the following geometric idea: Consider three circles (corresponding to tolerance circles) that are mutually external. It is desirable to find the curve passing within each circle that has the smallest possible value of its maximum of curvature. It is straightforward to show that such a curve is itself a circle. Furthermore, this circle is tangent to each of the tolerance circles. Similarly, the curve passing within each tolerance circle having the largest possible value for the minimum of curvature is also a circle tangent to each of the tolerance circles. An example for each type of constraint is shown in Figure 2-6.

Thus, for any set of three data points (whose tolerance circles do not overlap) it is possible to determine an upper bound for the minimum curvature of any curve passing through the tolerance circles. Similarly, it is also possible to determine a lower bound for the maximum curvature of a curve passing through the tolerance circles. By comparing the bounds of neighboring extrema, it is often possible to deduce that an extremum must be present. In this case, the extremum is verified.

More specifically, consider three neighboring extrema of curvature, a maximum that is adjacent to two minima. Note that each extremum has a critical data point associated with it, as described above. We assume for simplicity that these critical points are mutually external. In the proximity of the maximum, it is desirable to determine a lower bound for the maximum curvature. The lower bound circle is computed for the three tolerance circles of the extrema as illustrated in Figure 2-6a. Assume that the lower bound for the maximum curvature is given by $\kappa_{\max\ \mathrm{lb}}$.

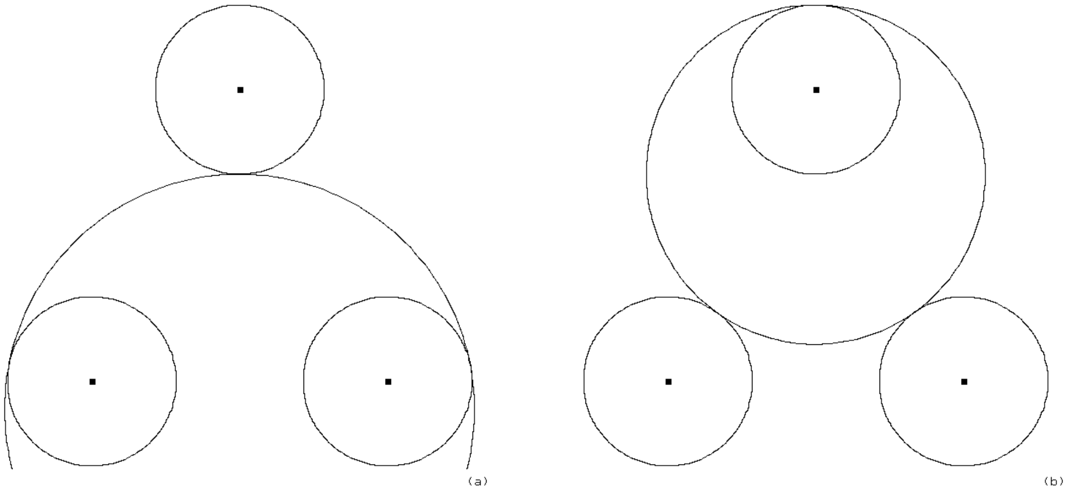


Figure 2-6: Determination of the upper bound of the minimum curvature and the lower bound of the maximum curvature. Three data points and their respective tolerance circles are depicted. Any curve passing within each of these tolerance circles has a maximum and minimum curvature, κ_{max} and κ_{min}, respectively. The curve having the lowest value for κ_{max} is the circle illustrated in (a). Thus, the curvature of this circle is a lower bound for the maximum curvature of a curve passing through each of the tolerance circles. Similarly, the curve having the largest possible value of κ_{min} is the circle illustrated in (b). The curvature of this circle is the upper bound for the minimum curvature of any curve passing through each of the tolerance circles.

Similarly, the upper bound for the minimum curvature for each of the minima of curvature may also be obtained. Assume that these values are given by $\kappa_{\text{min ub1}}$ and $\kappa_{\text{min ub2}}$, respectively. Under the condition that $\kappa_{\text{max}-\text{lb}} > \kappa_{\text{min}-\text{ub1}}$ and $\kappa_{\text{max}-\text{lb}} > \kappa_{\text{min}-\text{ub2}}$, a maximum of curvature must exist along the curve between the two critical points associated with the minima of curvature. A minimum of curvature may be verified similarly.

In the case where two consecutive extrema are nonessential, it is desirable to modify the curve further to eliminate these extrema. Consider two nonessential extrema and their respective neighboring essential extrema. Of course, one of the essential neighbors is a maximum and the other is a minimum. It is appropriate to increase the curvature of the neighboring essential maximum. Similarly, it is appropriate to decrease the curvature of the neighboring essential minimum. After doing this, it is possible to continue to deform the portion of the curve between the essential extrema to eliminate the nonessential extrema. The operation is analogous to backing out of a local minimum and resuming the original computation. An example of this operation is shown in Figure 2-7.

The algorithm to reduce the complexity of a contour coupled with the verification algorithm typically yields an instance of the optimal solution. However, when the

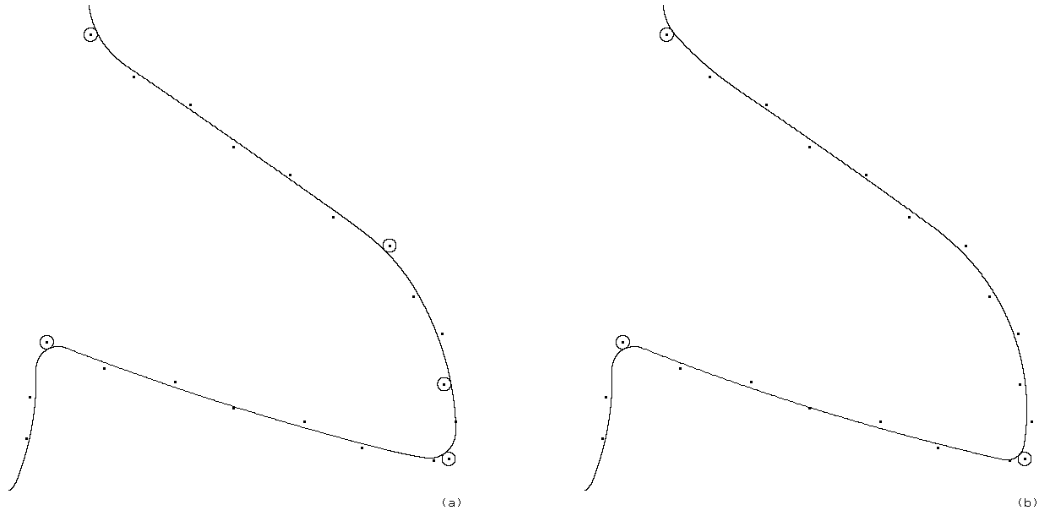


Figure 2-7: The elimination of an extrema pair with the aid of the verification technique. The curve in (a) is more complex than necessary given the scale parameter that is illustrated by the tolerance circles. However, the initial algorithm is unable to eliminate the unnecessary extrema pair. Upon application of the verification algorithm, it is determined that the maximum in the lower right is necessary while the two extrema just above it are unnecessary. The algorithm increases the curvature of the necessary maximum and proceeds. The algorithm is able to eliminate the unnecessary extrema. The result, a less complex curve, is shown in (b).

scale parameter, δ, is near a critical value, the algorithm may have difficulty finding the optimal solution. Consider an experiment where the value of δ is changed in very small increments. Typically, a small change in δ yields no change in the optimal value of the complexity measure, M. However, at particular values of δ, the optimal value of M decreases by a discrete amount (typical two). This occurs at a *critical value* of δ. When δ is equal to a critical value, a portion of the curve is uniquely defined in the section where the extrema were most recently eliminated. Thus, any arbitrarily small perturbation from the curve in this section of the curve would lead to a suboptimal result. Therefore, when δ is close to a critical value, the algorithm may fail to obtain an optimal solution.

This property of the algorithm would be troublesome if the "optimal" value of the scale parameter were chosen by some criterion. In this case, a small change in the "optimal" scale parameter would lead to significantly different results. Furthermore, if the "optimal" scale parameter were chosen to be close to a critical value of δ for some curve, the minimum complexity solution might not be found, thereby defeating the purpose of the "optimal" scale parameter. However, obtaining a multiple-scale representation of the contour alleviates these problems. If the algorithm fails to eliminate a non-essential pair of extrema at one scale, it is certain that the non-essential extrema will be eliminated in the next more coarse scale. Furthermore,

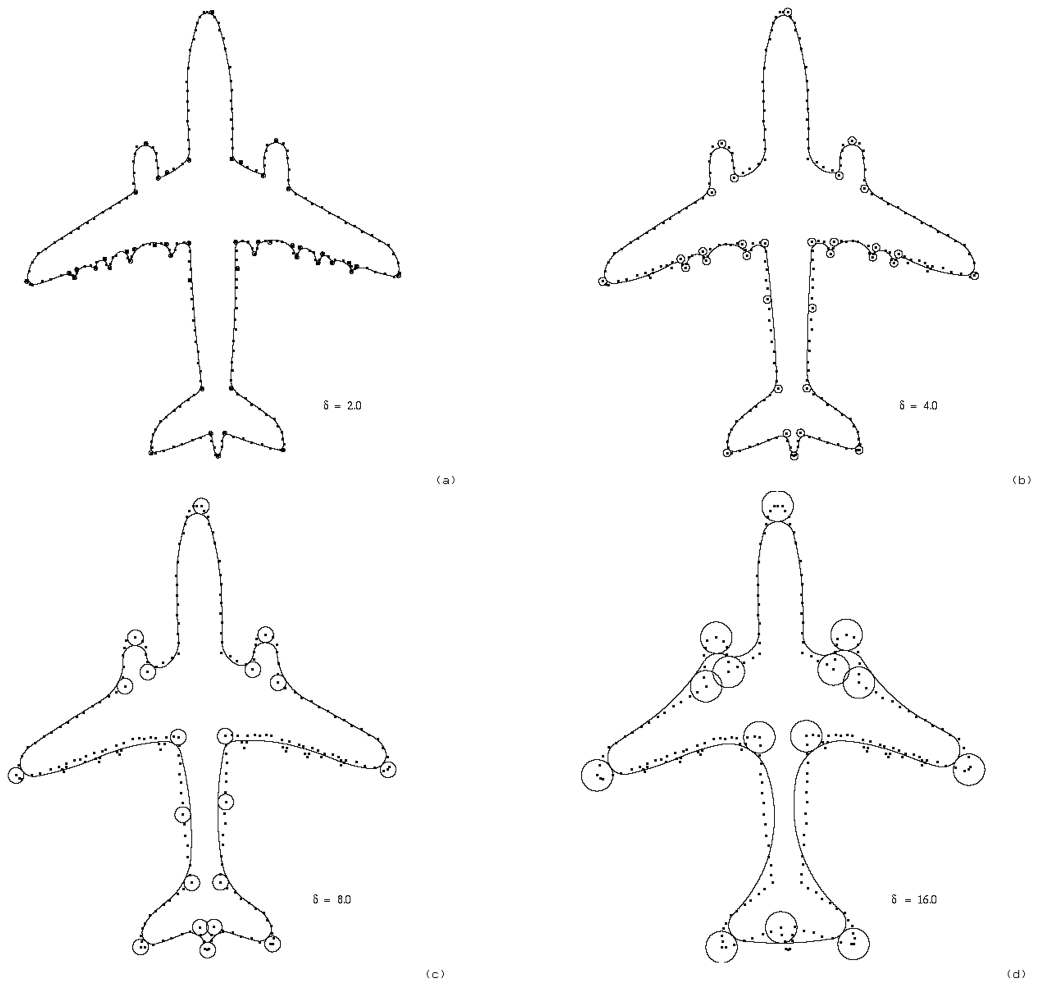


Figure 2-8: Minimum complexity curves for the silhouette of an airplane. Each curve is an instance of a minimum complexity curve for a particular tolerance value. The critical data points are indicated by the tolerance circle drawn about each such point. The radius of each of these circles is equal to δ, the scale parameter.

there is no sensitivity to the choice of the "optimal" scale near critical values of δ since no choice is made.

Example results of the computation of the minimum complexity curve for the silhouette of an airplane are shown in Figure 2-8. Each curve shown in the figure represents the output with a different scale parameter and, consequently, a different complexity measure. As the scale parameter increases, the details of the curve are lost and only the gross structure of the airplane remains.

2.4.3 Computation of the Least Square-Error Curve

An iterative procedure is used to compute the curve with least square-error under the constraint that the curve have the complexity determined in first stage. The curvature and position of each arc are modified locally until the change in the square-error of the curve from the data points tends to zero. The output of this stage is the desired result.

The least square-error curve is computed with the aid of a mechanical analogy. Under the analogy a frictionless spring is attached between each data point and the curve. The force exerted by the spring is linearly proportional to the distance between the point and the curve. The energy stored in each spring is the square of the distance. Hence, the total energy in the system is the square-error between the data points and the curve. We exploit this analogy to determine the appropriate deformation of the curve that reduces the energy of the system and analogously the square-error. The deformations are carried out under the constraint that they do not increase the complexity of the curve. The curve is deformed until it is no longer possible to reduce the square-error.

Rotation of Two Neighboring Arcs

Consider the deformation of the curve by rotating a pair of neighboring arcs. The torque on each of the arcs is computed analogous to the mechanical system described above. By combining the torques appropriately, it is possible to determine which direction the arc pair would be inclined to rotate. Once this is determined, the algorithm attempts to find new positions of the arcs that are consistent with the rotation calculated from the analogy. The position that most improves the square-error is chosen; the curve is updated appropriately.

As in section 2.3.3, consider two adjacent arcs, arc1 and arc2, and their respective neighbors, neighbor1 and neighbor2. The position of arc1 is modified by rotating it about the center point of neighbor1. The position of arc2 must also be modified to maintain tangency with arc1. Arc2 is also rotated; the center point of neighbor2 acts as the fulcrum for this rotation.

Under the mechanical analogy, each data point associated with a particular arc exerts a force on the arc proportional to the distance from the data point to the arc. The torque applied by a particular data point is the cross-product of the force applied by the data point and the lever arm. Positive torque corresponds to rotation in the clockwise direction about the fulcrum; negative torque corresponds to counterclockwise rotation. The sum of the torques applied by each data point yields the total torque acting on the arc by it data points.

Now consider the pair of arcs. Each has a set of data points acting to rotate the arc. Perhaps the forces act in such a way that the arcs rotate in a consistent direction, perhaps not. In the case where the torques oppose each other, it is necessary to determine which direction of rotation prevails. That is, it must be determined which direction of rotation would lead to a reduction in the local square-error of the curve.

To reconcile possibly differing directions of rotation, it is necessary to compute the effective force of one arc acting on the other. Without loss of generality, we consider the effective force of arc1 acting on arc2. Assume that the torque acting on arc1 by its associated data points is T_1. The force, F_e, that arc1 applies to arc2 is related to T_1 by

$$\mathbf{T_1} = \mathbf{r_1} \times \mathbf{F_e}, \tag{2.10}$$

where $\mathbf{r_1}$ is the vector given by

$$\mathbf{r_1} = \begin{vmatrix} x_{12} & - & x_{cn1} \\ y_{12} & - & y_{cn1} \end{vmatrix}, \tag{2.11}$$

where the point (x_{cn1}, y_{cn1}) is the center point of neighbor1 and (x_{12}, y_{12}) is the tangent point of arc1 and arc2. The direction of $\mathbf{F_e}$ is always along the line determined by the center points of arc1 and arc2. Therefore, if θ_{12} is the end angle of arc1 at the tangent point with arc2, $\mathbf{F_{e2}}$ may be written,

$$\mathbf{F_e} = F_e \begin{vmatrix} \cos\theta_{12} \\ \sin\theta_{12} \end{vmatrix}. \tag{2.12}$$

By combining equations 2.10 and 2.12, F_e, the scalar of the equivalent torque, may be written

$$F_e = \frac{\mathbf{T_1}}{\mathbf{r_1} \times \begin{vmatrix} \cos\theta_{12} \\ \sin\theta_{12} \end{vmatrix}}. \tag{2.13}$$

The total torque acting on arc2, $\mathbf{T_{t2}}$ may be written

$$\mathbf{T_{t2}} = \mathbf{T_2} + \mathbf{r_2} \times \mathbf{F_e}, \tag{2.14}$$

where the $\mathbf{r_2}$ is the lever arm for the effective force acting on arc2, $\mathbf{r_2}$, is given by

$$\mathbf{r_2} = \begin{vmatrix} x_{12} & - & x_{cn2} \\ y_{12} & - & y_{cn2} \end{vmatrix}. \tag{2.15}$$

If $\mathbf{T_{t2}}$ is positive, a small counterclockwise rotation of arc1 along with the appropriate rotation of arc2 leads to a decrease in the square-error. Conversely, if $\mathbf{T_{t2}}$ is negative, a small clockwise rotation of arc1 leads to a decrease in the square-error. Of course, if $\mathbf{T_{t2}}$ is zero, the arcs are in equilibrium with respect to rotation and rotation in either direction would increase the square-error. The geometric aspects of the mechanical analogy for rotation of two neighboring arcs are illustrated in Figure 2-9.

Once the appropriate direction of rotation has been determined, the algorithm finds the limiting case of rotation. That is, for some particular amount of rotation in

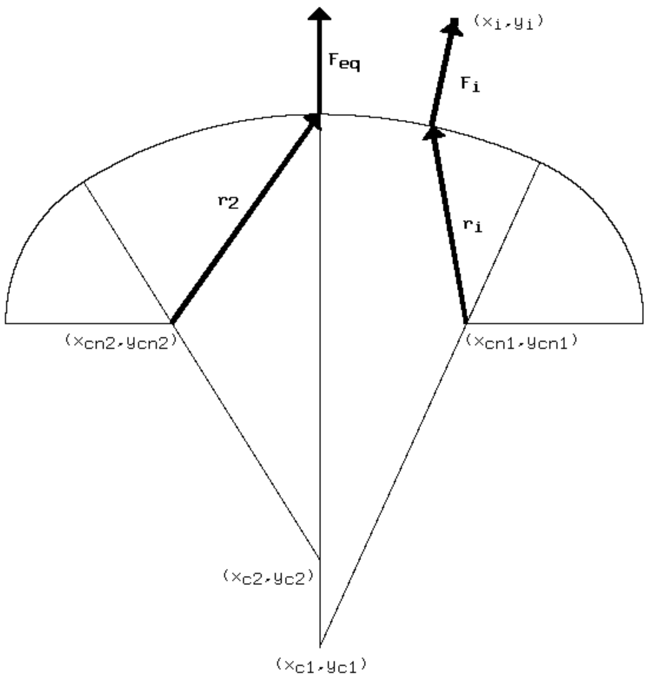


Figure 2-9: Torque acting to rotate neighboring arcs. Under the mechanical analogy, each data point applies a force to the curve. The resulting torque of data point (x_i, y_i) is the cross product of the force and the lever arm. Because the rotation of two neighboring arcs is coupled, it is necessary to compute the effective total force of one of the arcs acting on the other. The effective force is related to the total torque acting on the arc and the lever arm, $\mathbf{r_1}$.

the appropriate direction, the arclength of a particular arc becomes zero. This is the greatest extent to which the rotation may be carried out; further rotation would lead to an illegal configuration of arcs.

If the limiting case of rotation yields a reduction in the square-error of the curve, the rotation is chosen and the curve is updated appropriately. If the limiting rotation does not yield a reduction in the square-error, rotations of successively smaller extent are computed. If any rotation is found to reduce the square-error, it is chosen and the curve is updated appropriately. If no rotations are found to reduce the square-error, the curve is not modified by this operation.

More specifically, the rotation of two neighboring arcs has one degree of freedom. As described in Section 2.3.3, a convenient parameterization for the operation is the tangent point between neighbor1 and arc1. A particular position of the tangent point specifies the position of arc1. Given the position of arc1, the position of arc2 is specified, since arc2 must be tangent to both arc1 and neighbor1.

During the course of iteration, a particular tangent point between neighbor1 and arc1 may not lead to an improvement in the square-error. When this occurs, a new tangent point is chosen such that it is the midpoint between the previous tangent point and the original tangent point. In this way the extent of rotation is cut in half at each attempt. When the attempted tangent point is within a specified tolerance, ϵ, of the original tangent point, no more rotations are attempted and the curve is not modified.

Modification of the Curvature of a Single Arc

Consider the deformation of the curvature of a single arc. The derivative of the energy in the system with respect to the curvature of the arc is the total deformation force acting on the arc. If the derivative of energy with respect to curvature is positive, the curvature of the arc is decreased. Conversely, if the derivative of energy with respect to curvature is negative, the curvature is increased. The curve is updated appropriately.

Consider an arc with a single data point, p_i. The force acting to deform the curvature of the arc is equal to the distance from the point to the arc. The force vector $\mathbf{F_i}$ is given by

$$\mathbf{F_i} = \left(\sqrt{(x_i - x_c)^2 + (y_i - y_c)^2} - R\right) \left| \begin{array}{c} \cos\theta_i \\ \sin\theta_i \end{array} \right|, \tag{2.16}$$

where θ_i is the angle of the vector from the center of the arc to the $\mathrm{i^{th}}$ data point. The energy related to the $\mathrm{i^{th}}$ point is given by

$$E_i = \left[\sqrt{(x_i - x_c)^2 + (y_i - y_c)^2} - R\right]^2. \tag{2.17}$$

The modification of the curvature of the arc occurs under the constraint that the arc must remain tangent to both of its neighbors. For small perturbations of the curvature, the effect of this constraint is approximated by requiring the arc to remain tangent to the two lines that are each tangent to the arc and one of its neighbors. Under this approximation, the center of the arc is constrained to lie on the line defined by the original center point of the arc and the intersection of the two tangent lines. Figure 2-10 illustrates this geometric constraint.

The scalar force acting on the center of the arc, F_c, is the derivative of the energy with respect to motion along the constraint line. Let $d_i = \sqrt{(x_i - x_c)^2 + (y_i - y_c)^2}$. The force may be written,

$$F_c = \frac{dE}{ds} = 2\,(d_i - R)\left[\frac{dd_i}{ds} - \frac{dR}{ds}\right], \tag{2.18}$$

where s is the scalar distance along the constraint line. The radius of the circle

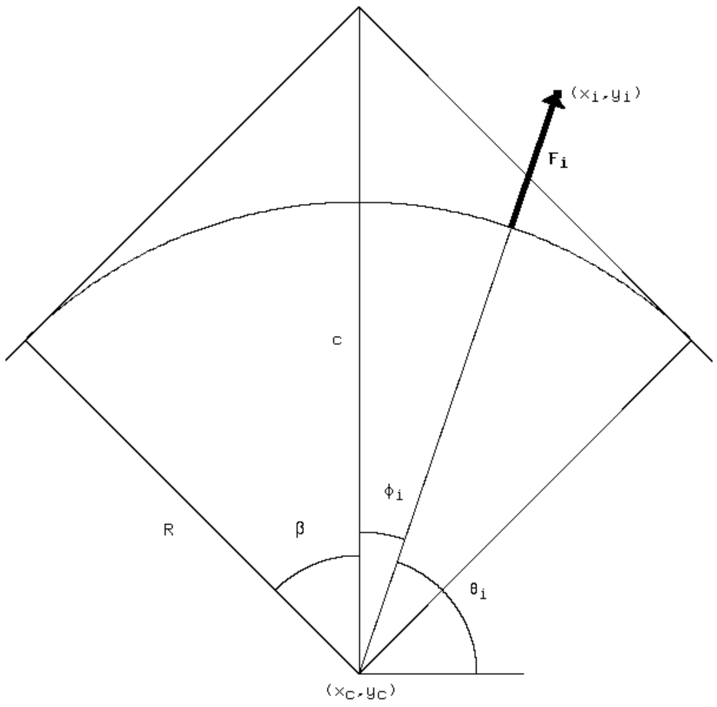


Figure 2-10: The force acting to deform the curvature of a single arc. The force acting on the arc by each associated data point is equal to the distance from the point to the arc. For small perturbations of the curvature, the constraint that the arc must remain tangent to its neighbors is approximated by requiring the arc to remain tangent to each line tangent to the arc and one of its neighbors.

may be written as $R = c\cos\beta$, where β is half the angular width of the arc and c is the distance between the center of the circle and the intersection of the tangent lines. Conveniently, $\frac{dc}{ds} = 1$ and $\frac{dR}{ds} = \cos\beta$. Similarly, $\frac{dd_i}{ds} = \cos\phi_i$, where ϕ_i is the angle between the constraint line and the data point. Thus, Equation 2.18 may be simplified

$$F_c = 2\left(d_i - R\right)\left(\cos\phi_i - \cos\beta\right) = 2F_i\left(\cos\phi_i - \cos\beta\right). \tag{2.19}$$

In the derivation of Equation 2.19, we have tacitly assumed that the angle of the sector of the arc is less than π. If the angle is greater than π, the force acting on the center of curvature is given by

$$F_c = -2F_i\left(\cos\phi_i - \cos\beta\right). \tag{2.20}$$

When two or more data points are associated with the arc, the total force acting on the center of the circle is the sum of the individual forces associated with the individual points. If the total force is positive, then the center point is inclined to move along the constraint line toward the intersection point of the tangent lines. That is, a small perturbation of the center toward the intersection of the tangent lines leads to a reduction of the local square-error of the curve. If the total force is negative, the center point is inclined to move in the opposite direction.

From the direction of preferred motion, it is possible to determine whether the arclength of the neighbors will increase or decrease. Equivalently, it is possible to determine the direction of motion of the tangent points of the neighbors. Because the computation of the deformation of curvature is parameterized by the location of the tangent point with one of the neighbors, this provides enough information to compute candidate arcs.

There is a particular deformation of the arc that leads to an arclength of zero for one of the neighbors. This is the limiting case of legal deformations of curvature. If this deformation would lead to a reduction in the square-error of the curve, it is chosen and the curve is updated appropriately. If the deformation does not lead to a decrease in the square-error, successively less severe deformations are attempted. On each iteration, the new deformation is chosen to be midway between the location of the previous tangent point and the original tangent point. If any of the deformations reduces the square-error of the curve, it is chosen. If not, the iteration is halted when the candidate tangent point is within a distance ϵ of the original tangent point.

Splitting an Arc into Two Arcs

Consider the deformation of an arc by splitting it into two arcs. This is necessary when data points in one segment of the arc are acting to increase the curvature of the arc while in another segment the data points are acting to decrease the curvature. The ability to split an arc provides additional degrees of freedom that make it possible to further reduce the square-error. Again, we refer to the new arcs as arc1 and arc2 and to their respective neighbors as neighbor1 and neighbor2.

It is necessary to determine the appropriate location of the break in the original arc. This is accomplished by considering a number of candidate break points along the arc and choosing the best location in the sense described below. The candidate split locations are chosen such that they lie at the midpoint of each pair of neighboring projections of data points onto the arc.

At each candidate split point a constraint line is constructed such that it is tangent to the original arc. Conceptually, the arc is broken into two sub-arcs that have the same curvature. For each sub arc, the curvature deformation force is computed as described in Section 2.4.3. That is, the tendency of each sub-arc to increase or decrease curvature is determined. If the sub-arcs have consistent tendencies then it is not appropriate to split the arc at that point. However, if one of the sub-arcs has a total force acting to increase its curvature while the other sub-arc has a force acting to decrease its curvature, splitting the arc would lead to a decrease in the square-error of the curve. This condition is illustrated in Figure 2-11.

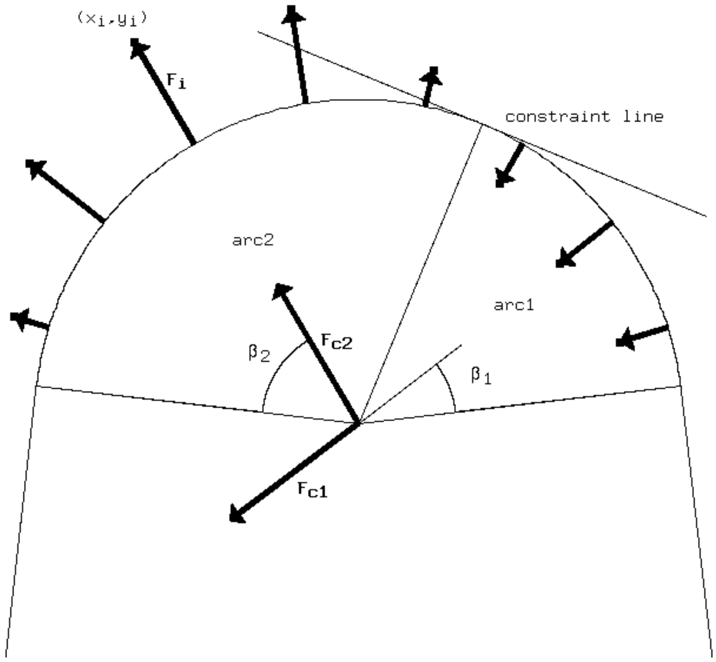


Figure 2-11: Setup of the operation to split an arc into two. By splitting an arc into two, it is possible to increase the curvature of one of the new arcs while decreasing the curvature of the other. For the choice of the constraint line shown, the forces are acting to decrease the curvature of arc1 and increase the curvature of arc2.

There may be multiple candidate break points that would lead to a decrease in the square-error of the curve. The candidate break point is chosen that minimizes the derivative of square-error with respect to the position of the tangent point between arc1 and arc2 along the constraint line. This measure is chosen because the deformation is ultimately parameterized by this location.

We define S, the tendency of an arc to split, as the derivative of the energy with respect to the position of tangency along the constraint line. That is,

$$S = \frac{dE}{ds}, \tag{2.21}$$

where s is a parameter that specifies the position of the tangent point of arc1 and arc2 along the constraint line. S may be written

$$S = F_{c1} \sin \beta_1 + F_{c2} \sin \beta_2, \tag{2.22}$$

where F_{c1} and F_{c2} are the respective forces acting on the centers of each sub-arc and β_1 and β_2 are the respective half-angles of the sub-arcs. The expression on the right side of Equation 2.22 is a result of the chain rule applied to Equation 2.21. As described in Section 2.4.3, F_c is the derivative of energy with respect to position of the center point. Here, $\sin\beta$ is the derivative of the position of the center point with respect to the location of the position along the tangency constraint line.

Once the break point has been chosen, the arc is deformed as described in Section 2.3.3. The limiting case of the deformation is computed explicitly. That is, there is a particular deformation that leads to a zero arclength for one of the neighbors. If the limiting case results in a reduction of the square-error, the deformation is chosen and the curve is updated appropriately. Otherwise, successively smaller deformations are chosen until one is found that improves the square-error. As above, the iteration is terminated when the candidate tangent point is within ϵ of the original tangent point.

The results of this stage applied to the silhouette of an airplane are shown in Figure 2-12. Notice that as the complexity of the curve decreases, the fine details of the silhouette are lost. However, the global structure of the airplane remains intact.

Discussion

The least square-error curve is computed by iterating over the arcs of the curve. The operations described above are applied to each arc. For each operation, the square-error is guaranteed to decrease or remain the same. The algorithm stops when the curve is not modified on a particular cycle or when the decrease in square-error is negligible.

For each operation, the constraints acting on each arc are highly non-linear. Thus, it is not possible to ensure that the optimal solution is always found. In practice, however, it is likely that the algorithm will provide a solution that is close to optimal.

The energy function, defined above, may be viewed as a Liapunov function when the computation is sufficiently close to the optimal solution (see, for example, Luenberger[50]). If the energy is not allowed to increase, the state of computation remains within a finite neighborhood of the optimal solution. Furthermore, decreasing the energy always yields a smaller neighborhood about the optimal.

Therefore, if the initial state of the computation is close enough to the optimal solution, the optimal solution will be found. The output of the first stage of the computation is a natural starting curve. The curve is constrained to be close (within δ) to each data point. Hence, it is likely that the algorithm typically produces reasonable results.

There is room for improvement in the algorithm, however. It is possible to deform any arbitrary curve into any other arbitrary curve using the deformations described above. However, it is not guaranteed that there is a path from an initial state to the optimal state such that each step along the path yields a non-increasing energy function. There are alternative deformations that would provide additional options

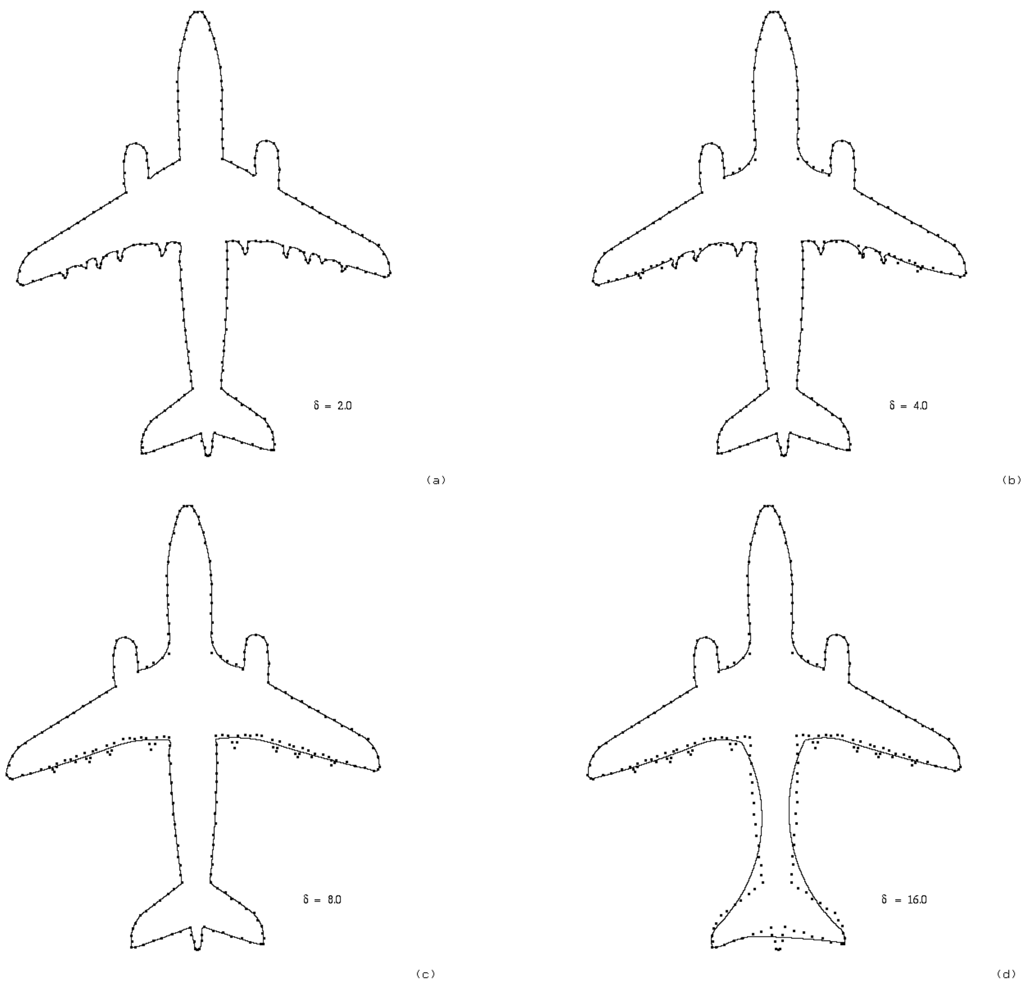


Figure 2-12: Minimum complexity/least-square-error curves for the silhouette of an airplane. In each case the least-square error curve that has the complexity determined from the first stage of the computation is shown. The scale parameter for each case is (a) $\delta = 2.0$, (b) $\delta = 4.0$, (c) $\delta = 8.0$, and (d) $\delta = 16.0$.

at each step of the computation. The implementation of such options is likely to reduce the frequency that the algorithm finds local minima rather than the optimal solution.

2.5 Mathematical Properties of the Contour

In this section we describe the computation of a number of mathematical properties associated with contours.

2.5.1 Point Properties of the Contour

The curve is represented as a list of N circular arcs. The i^{th} arc is specified by its center (x_{ci}, y_{ci}), the radius, R_i, the curvature, κ_i, and two end angles, θ_{1i} and θ_{2i}. The i^{th} arc is always tangent to the $i+1^{th}$ arc.

The coordinates of points on the curve may be described parametric in arclength, s, as

$$x(s) = \begin{cases} x_{c1} & + \quad R_1 \cos\left(s\kappa_1 + \theta_{11}\right) \qquad 0 < s < s_1, \\ x_{c2} & + \quad R_2 \cos\left(\left(s - s_1\right)\kappa_2 + \theta_{12}\right) \qquad s_1 < s < s_2, \\ & \vdots \\ x_{cN} & + \quad R_N \cos\left(\left(s - s_{N-1}\right)\kappa_N + \theta_{1N}\right) \quad s_{N-1} < s < s_N. \end{cases} \tag{2.23}$$

Similarly,

$$y(s) = y_{ci} + R_i \sin\left(\left(s - s_{i-1}\right)\kappa_i + \theta_{1i}\right) \qquad s_{i-1} < s < s_i. \tag{2.24}$$

The orientation of the curve at a particular point is specified by the normal vector at that point. The normal vector is perpendicular to the tangent to the curve and points away from the interior of the curve. The normal vector points in the same direction as the radius of the circular arc when the curvature is positive. The normal vector points in the opposite direction of the radius when the curvature is negative. The unit normal vector for any point on the i^{th} arc is given by

$$\hat{\mathbf{n}}_{\mathbf{i}}(s) = \operatorname{sgn}\left(\kappa_i\right) \begin{vmatrix} \cos\left(\left(s - s_{i-1}\right)\kappa_i + \theta_{1i}\right) \\ \sin\left(\left(s - s_{i-1}\right)\kappa_i + \theta_{1i}\right) \end{vmatrix} \qquad s_{i-1} < s < s_i. \tag{2.25}$$

2.5.2 Points Interior and Exterior to the Contour

The determination of whether a point is in the interior or the exterior of a curve is a global operation. That is, the entire curve must be considered. Given the curve representation described here, it is straightforward to determine the relations of the point to the curve.

It is desirable to find the closest arc of the curve to the point. The point of interest is guaranteed to be within the sector of the closest arc. This follows from the definition of the distance from an arc to a point and the constraint that the arcs are mutually tangent. Two cases must be considered for the computation of the distance from a point to an arc. If the point is within the sector of the arc, the distance to the arc is simply the distance to the circle. If the point is not within the sector, the distance to the arc is the minimum of the distances to each of the endpoints.

If the closest arc has positive curvature and the point is in the interior of the circle associated with the arc, the point is in the interior of the curve. If the closest arc has positive curvature and the point is in the exterior of the circle, the point is in the exterior of the curve. Conversely, if the closest arc has negative curvature, the relations are reversed (interior of the circle implies exterior of the curve and vice versa).

2.5.3 Perimeter and Area

The perimeter of the region bounded by the curve may be computed. The perimeter is equal to the sum of the arclengths of the individual arcs. The perimeter, P, may be expressed

$$P = \sum_{i=1}^{N} R_i \left|\theta_i - \theta_{i-1}\right|. \tag{2.26}$$

The area of the interior of the curve may be computed by applying Green's Theorem[71]. With appropriate application of Green's Theorem, the area, A, may be computed

$$A = \iint_R dx\, dy = \frac{1}{2} \oint_C (x\, dy - y\, dx). \tag{2.27}$$

This integral may be expressed

$$A = \sum_{i=1}^{N} A_i = \sum_{i=1}^{N} \frac{1}{2} \int_{s=s_{i-1}}^{s=s_i} \left(x(s) \frac{\partial y}{\partial s}\, ds - y(s) \frac{\partial x}{\partial s}\, ds \right), \tag{2.28}$$

where each term of the sum is an integral over a single circular arc. The i^{th} term of this sum is simply an integral over a single circular arc. By suitable change of variable, the integral may be written,

$$A_i = \frac{1}{2} \int_{\theta_{1i}}^{\theta_{2i}} \left(x(\theta) \frac{\partial y}{\partial \theta}\, d\theta - y(\theta) \frac{\partial x}{\partial \theta}\, d\theta \right), \tag{2.29}$$

where θ is the angular displacement of the arc. Making appropriate substitutions, each term may be written

$$\begin{aligned} A_i &= \frac{1}{2} \int_{\theta_{1i}}^{\theta_{2i}} \left((x_c + R\cos\theta)(R\cos\theta) - (y_c + R\sin\theta)(-R\sin\theta) \right) d\theta, && (2.30) \\ &= \frac{1}{2} \left\{ R x_c \int_{\theta_{1i}}^{\theta_{2i}} \cos\theta\, d\theta + R^2 \int_{\theta_{1i}}^{\theta_{2i}} \left(\sin^2\theta + \cos^2\theta \right) d\theta + R y_c \int_{\theta_{1i}}^{\theta_{2i}} \sin\theta\, d\theta \right\}, && (2.31) \\ &= \frac{1}{2} \left\{ R x_c \left[\sin\theta\right]_{\theta_{1i}}^{\theta_{2i}} - R y_c \left[\cos\theta\right]_{\theta_{1i}}^{\theta_{2i}} + R^2 \left[\theta_{2i} - \theta_{1i}\right] \right\}. && (2.32) \end{aligned}$$

The sum over all of these terms yields a closed form expression for the area of the interior of the curve.

2.5.4 Centroid

Following the derivation for the area computation, we may compute the centroid of a region directly from the contour representation. The centroid of a region is given by

$$\begin{aligned} \overline{x} &= \frac{1}{A}\iint_R x\,dx\,dy, & (2.33)\\ \overline{y} &= \frac{1}{A}\iint_R y\,dx\,dy. & (2.34)\end{aligned}$$

Again, applying Green's Theorem, we find

$$\begin{aligned} \overline{x} &= \frac{1}{A}\oint_C (-xy)\,dx, & (2.35)\\ \overline{y} &= \frac{1}{A}\oint_C (xy)\,dy. & (2.36)\end{aligned}$$

For convenience, we first consider the computation of the x coordinate of the centroid.

The integral on the right side of Equation 2.35 may be expressed

$$\oint_C (-xy)\,dx = \sum_{i=1}^{N} B_i = \sum_{i=1}^{N}\int_{s=s_{i-1}}^{s=si}[-x(s)y(s)]\,\frac{\partial x}{\partial s}\,ds, \tag{2.37}$$

where each term of the sum is an integral over a single circular arc. The i^{th} term of this sum is the integral over a single circular arc. By suitable change of variable, the integral may be written,

$$B_i = \int_{\theta_{1i}}^{\theta_{2i}}[-x(\theta)\,y(\theta)]\,\frac{\partial x}{\partial\theta}\,d\theta, \tag{2.38}$$

where θ is again the angular displacement of the arc. Making appropriate substitutions, each term may be written

$$\begin{aligned} B_i &= \int_{\theta_{1i}}^{\theta_{2i}}\left[-(x_c+R\cos\theta)(y_c+R\sin\theta)(-R\sin\theta)\right]\,d\theta, & (2.39)\\ &= R\int_{\theta_{1i}}^{\theta_{2i}}\left(x_cy_c+x_cR\sin\theta+y_cR\cos\theta+R^2\sin\theta\cos\theta\right)\sin\theta\,d\theta, & (2.40)\\ &= Rx_cy_c\int_{\theta_{1i}}^{\theta_{2i}}\sin\theta d\theta+R^2x_c\int_{\theta_{1i}}^{\theta_{2i}}\sin^2\theta d\theta+ & \\ &\quad R^2y_c\int_{\theta_{1i}}^{\theta_{2i}}\sin\theta\cos\theta d\theta+R^3\int_{\theta_{1i}}^{\theta_{2i}}\sin^2\theta\cos\theta d\theta, & (2.41)\end{aligned}$$

$$= -Rx_cy_c\left[\cos\theta\right]_{\theta_{1i}}^{\theta_{2i}} + R^2x_c\left[\frac{\theta}{2}-\frac{\sin 2\theta}{4}\right]_{\theta_{1i}}^{\theta_{2i}} - R^2y_c\left[\frac{\cos 2\theta}{4}\right]_{\theta_{1i}}^{\theta_{2i}} + R^3\left[\frac{\sin^3\theta}{3}\right]_{\theta_{1i}}^{\theta_{2i}}. \qquad (2.42)$$

Using this expression the x coordinate of the centroid may be written

$$\overline{x} = \frac{1}{A}\sum_{i=1}^{N} B_i. \qquad (2.43)$$

Following a parallel argument, the y coordinate of the centroid may be written

$$\overline{y} = \frac{1}{A}\sum_{i=1}^{N} C_i, \qquad (2.44)$$

where

$$C_i = Rx_cy_c\left[\sin\theta\right]_{\theta_{1i}}^{\theta_{2i}} - R^2x_c\left[\frac{\cos 2\theta}{4}\right]_{\theta_{1i}}^{\theta_{2i}} + R^2y_c\left[\frac{\theta}{2}+\frac{\sin 2\theta}{4}\right]_{\theta_{1i}}^{\theta_{2i}} - R^3\left[\frac{\cos^3\theta}{3}\right]_{\theta_{1i}}^{\theta_{2i}}. \qquad (2.45)$$

Thus, we have obtained a closed form expression for the centroid of a region.

2.5.5 Extended Circular Image

The extended circular image has been proposed as a useful description of the shape of a curve[42]. The value of a particular point of the extended circular image, $C(\psi)$, for a convex curve is

$$C(\psi_a) = \frac{1}{\kappa(s_a)}, \qquad (2.46)$$

where ψ_a refers to a particular orientation angle and $\kappa(s_a)$ is the curvature of the curve at the location where the unit normal vector makes an angle ψ_a with the x-axis. If the curve is not convex, more than one point of the curve maps onto a particular point of the extended circular arc. In that case, the value of the extended circular image is the sum of the curvatures of all points on the curve with the appropriate orientation.

The value of a single point on the extended circular image is easily determined from the curve representation. All points on the contour with the appropriate orientation must be found. The value of the extended circular arc is the sum of the reciprocal of the curvature associated with each of these points.

The extended circular image may be computed as a function as well. A single arc of the curve contributes a constant value (the value of the reciprocal of its curvature) over a particular range of orientations. That range of orientations is, of course, the range of orientations of the arc. Consequently, the extended circular image for any curve represented by piecewise circular arcs is piecewise constant. The break points of the piecewise constant function are the orientations at the curve points where neighboring arcs are tangent.

2.6 Summary

In this chapter, we consider an analytical representation of contours. A contour is represented as a list of pairwise tangent circular arcs. The representation leads to improved computation of mathematical properties of the contour such as the curvature, orientation, bounded area, and centroid. We present a novel approach to contour smoothing. The complexity, rather than the magnitude of curvature, is employed in the smoothness criteria. We propose a scale-space based on the complexity measure. This space is truly a scale-space rather than a resolution-space. As we shall see, the improved representation and smoothing capability for contours facilitates more robust and reliable capabilities for higher level processing.

Chapter 3

The Medial Axis Skeleton

3.1 Introduction

The medial axis skeleton is a thin line graph that preserves the topological relationships of salient parts of a region. The skeleton has often been cited as a useful representation for shape description, region interpretation, and object recognition. The skeleton provides a decomposition of the region into salient subparts. It also provides a description of the connectivity of the subparts.

Unfortunately, the computation of the skeleton is extremely sensitive to variations in the bounding contour of a region. Tiny perturbations in the contour often lead to spurious branches of the skeleton. It is non-trivial to determine which of the branches are spurious and which correspond to significant subregions.

There have been numerous attempts to find a robust algorithm for computing the medial axis skeleton (see, for example, [5], [8], [19], [23], [35], [47], [54], [81]). Most algorithms use some deviation of morphological thinning. Often, spurious branches are eliminated based upon some approximate property of the bounding contour, or based upon some property of the branch itself.

One common problem with previous approaches is that the resulting skeleton is inconsistent with the bounding contour or the region from which it was computed. Inconsistencies between the representations of the skeleton and the contour may lead to inconsistent inferences in higher level processes. If such inconsistencies could be eliminated, the performance of higher level processes would be improved.

Another problem with previous approaches is that the results are typically mediocre. Most algorithms are only capable of handling simple objects like pseudopods, for example[47]. These algorithms fail because they are incapable of distinguishing between "noise" in the data and subtle features that may exist on the contour. As a result, most algorithms tend to produce skeletons with spurious branches, or they tend to provide skeletons that are unduly simplified.

Because computation of the skeleton is so sensitive, it is desirable to represent the skeleton across a variety of scales. The multiple scale description eliminates the need to determine the "optimal" scale by some artificial means. Attempts to find an optimal scale parameter in this and other contexts typically yield only marginal results.

Ideally, a scale-space for the medial axis skeleton would provide representations of the skeleton with varying levels of detail. At finer scales, the skeleton would have a larger number of branches; a greater number of features would be represented. At more coarse scales, the skeleton would have fewer branches; the skeleton would represent only the gross structure of the region.

The key to obtaining a multiple scale representation for the skeleton is to determine which branches should be eliminated as the algorithm moves from fine to coarse scales. Furthermore, it is necessary to determine the appropriate position of the skeleton branches so that they accurately depict the structure of the region. Finally, it is desirable to modify the bounding contour of the region, simultaneously, so that each skeleton in the scale-space corresponds to a consistent bounding contour.

In this paper, we consider a robust method for computing the medial axis skeleton across a variety of scales. The scale-space is parametric with the complexity of the skeleton. The complexity measure is defined as the number of branches of the skeleton.

The complexity of the skeleton is related to the complexity of the bounding contour. The complexity of the contour is measured by the number of extrema of curvature contained in the contour[15]. As we shall see, there is a formal relationship between the complexity measure of the contour and that of the skeleton. Thus, minimizing the contour complexity is tantamount to minimizing the skeleton complexity.

A set of curves is computed to represent the bounding contour across a variety of complexity measures. The curves possessing larger complexity measures represent greater detail than curves with smaller measures. A medial axis skeleton is computed directly from each contour. The result is a set of skeletons that represent only the gross structure of the region at coarse scales (low complexity), but they represent more of the detail at fine scales (high complexity).

In Section 3.2, we discuss the concept of complexity in greater detail. In Section 3.3, we consider the computation of the medial axis skeleton directly from the bounding contour. In Section 3.4, we define a scale-space for the medial axis skeleton that is based on the complexity measure.

3.2 Complexity

The complexity of an object may be viewed as the number of primitive components of the object. Similarly, the complexity of a representation of an object may be measured by the number of subparts contained within the representation. In this section we seek to formalize this notion of complexity for contours and the medial axis skeleton.

Hoffman and Richards[36] have proposed the use of *codons* to decompose a contour into salient parts. They observe that minima of curvature of a contour serve as natural break points of the curve. Therefore, the curve is broken into sections that are bounded by extrema of curvature. These sections are called codons. Pairs of codons typically correspond to subjective parts of the region bounded by the contour.

Given this insight, the number of codons contained in the contour is a reasonable measure of the number of subjective features of the region bounded by a contour. Therefore, the number of codons contained in the contour is a suitable measure of the complexity of the curve. Conveniently, the number of codons contained in the contour is equal to the number of extrema of curvature of a closed contour.

Similarly, the complexity measure of the medial axis skeleton is the number of branches of the skeleton. Each branch of the skeleton corresponds to a subregion of the region bounded by the contour. A region possessing a larger number of subregions is more complex than a region possessing a smaller number of subregions.

The complexity measure for contours is related to the complexity measure of the medial axis skeleton. Each branch of the skeleton that terminates into the contour, rather than into a node of the skeleton, does so at a positive maximum of curvature. Therefore, an upper bound for the number of branches in the skeleton may be obtained from the number of extrema of curvature of the bounding contour. If the number of extrema of a particular curve is M, then there are at most $M/2$ positive maxima of curvature. Therefore, there are no more than $M/2$ branches that terminate into the contour. Each of these terminal branches intersects another branch at a node and a third branch emanates from the node. This branch may or may not be a terminal branch. In the worst case, the number of non-terminal branches is $M/2-2$. Therefore, the complexity of the skeleton, B, is no larger than $M-2$.

A similar argument may be made for a region with holes. First, consider the skeleton associated with the bounding contour without holes. From above, the skeleton associated with the bounding contour has complexity $M-2$. Now add the holes one by one and consider the resulting skeleton. Each time a hole is added, the number of branches increases by no more than three. Thus, the maximum number of skeleton branches for a region with holes is M + 3H -2, where H is the number of holes.

More importantly, if the bounding contour is deformed continuously in such a way that the complexity measure decreases, the complexity measure of the corresponding skeleton almost always decreases. Equivalently, reducing the number of extrema of curvature of the bounding contour almost always causes the number of branches of the skeleton to decrease.

There is a tradeoff between the descriptive power of a representation and the complexity of the representation. If the complexity is allowed to be arbitrarily large, any set of data may be represented. On the other hand, limiting the complexity restricts the class of shapes and objects that may be represented. In Section 3.4, we exploit this tradeoff to define a scale-space for the medial axis skeleton that is similar to a complexity scale-space for contours. In the next sections, we consider the computation of the medial axis skeleton directly from the representation.

3.3 Computation of the Medial Axis Skeleton

The medial axis skeleton may be computed directly from the analytical contour representation. In this section, we consider the mechanics of the computation. First, we consider a number of useful general properties of the skeleton and its bounding contour. Next, we consider properties of the skeleton when the contour is made up of pairwise tangent circular arcs. Finally, we consider the computation of the skeleton from the analytical contour representation.

The medial axis skeleton is usually defined as the locus of points where wavefronts propagating inward from the bounding contour meet (see, for example, [47]). The skeleton points are locations where two or more wavefronts have propagated the same distance from their respective starting locations. This definition suggests morphological operators that approximate the propagation of the wavefront.

The medial axis skeleton may also be defined by the following properties: Each point on the skeleton is equidistant from two or more points on the bounding contour. There are no points on the boundary closer to the skeleton point than these equidistant points. And, each skeleton point lies in the interior of the bounding contour.

This alternate definition is mathematically equivalent to the wavefront definition. The distance from each skeleton point to the closest points on the contour is the distance traveled by the associated wavefronts. As we shall see, the alternate definition is constructive; it leads to a novel method of computing the skeleton.

Each point that is on a branch of a skeleton, but not a node, is equidistant from exactly two points on the contour. Each node point is equidistant from three or more points on the bounding contour. Typically, a node is equidistant from exactly three boundary points. The case where the node is equidistant from more than three points is a zero measure condition.

For each point on the branch of the skeleton there is a circle that is tangent to the contour in two places. The center of the circle is coincident with the point on the branch. The radius of the circle is the distance from the center to the two nearest points on the contour. Aside from the two tangent points, the circle does not contact the contour. We call such a circle the *interior circle* of the point of interest. We call the two points of tangency between the interior circle and the contour the *tangent points* of the interior circle.

Similarly, for each node point, there is a circle that is tangent to three (or more) points on the contour. The center of the circle is coincident with the node point and the radius is the distance from the node to the three nearest points on the contour. Aside from these tangent points, the circle does not contact the contour. Such a circle is called the *interior circle* of the node.

As a branch of the skeleton is traversed, the radius of the interior circle varies continuously. Stated another way, the distance from a skeleton branch to the contour varies continuously along the branch. Furthermore, as the branch is traversed, each tangent point of the interior circle moves continuously along the contour. In the

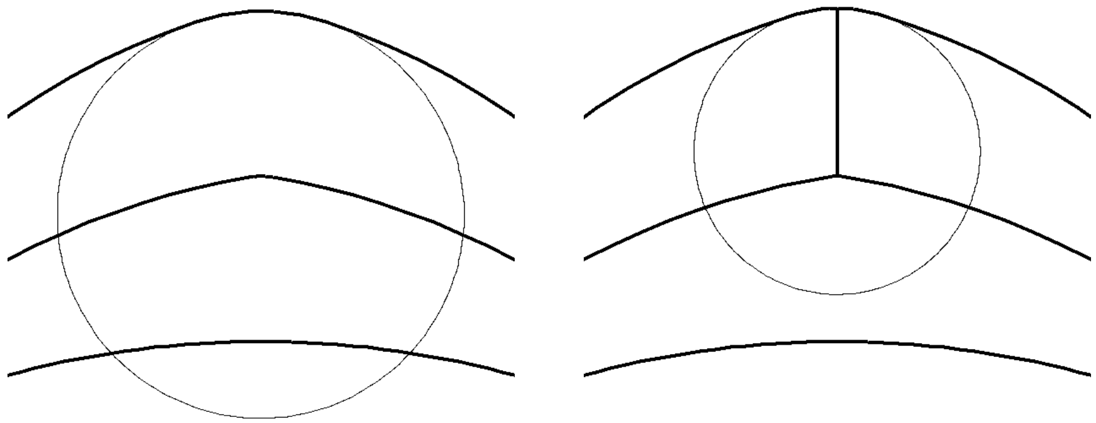


Figure 3-1: Test for the existence of branch terminating into a maximum of curvature. Each illustration depicts a portion of a contour and its medial axis skeleton; the osculating circle for the maximum of curvature at the top of the curve is also shown. In (a), no branch terminates into maximum of curvature because the osculating circle extends to the exterior of the curve. In (b), a branch does terminate into the maximum of curvature because the osculating circle remains in the interior of the contour.

case where the branch terminates into the contour, rather than into a node, the two tangent points converge with the branch at a point of maximum curvature on the contour.

Each branch of the skeleton that terminates into the contour does so at a positive maximum of curvature. It is not the case, however, that each positive maximum of curvature is associated with a branch termination. There is a simple test to determine if a positive maximum is associated with the termination of a branch. If the osculating circle associated with the curvature maximum lies completely in the interior of the contour or on the contour, there is a branch that terminates at the maximum. Otherwise, there is no terminus. The terminus test is illustrated in Figure 3-1.

Now, consider a contour that consists of pairwise tangent circular arcs, as described in Chapter 2. At any point on the skeleton, the tangent points lie on two particular arcs of the contour representation. Locally, the points on the skeleton branch are equidistant from these two arcs. A locus of points that is equidistant from two circles is a conic section. Therefore, the medial axis skeleton consists of segments of curves that are conic sections. We call such a curve segment a *conic segment* of the skeleton branch. The analytical contour representation leads directly to an analytic representation for the medial axis skeleton.

Because the conic segments of the skeleton are well characterized, it is convenient to compute the skeleton in a piecewise fashion. The key to computing this representation is finding the end points of the branch segments. At an end point of a branch segment, one of the tangent points of the interior circle is guaranteed to be coincident with the point of tangency of two neighboring contour arcs. Therefore, the segment end point must lie on the line determined by the radius of the circle corresponding to the end angle of the arc.

Assume that in some intermediate stage of the computation, a branch segment end point has been found. The arcs associated with the next branch segment are called arc1 and arc2, arbitrarily. The line determined by the center of arc1 and the end point of arc1 is called line1. The line determined by the center of arc2 and the end point of arc2 is called line2. Assume, without loss of generality, that the branch segment is hyperbolic. There are two possibilities for the location of the next segment end point. The end point may coincide with the intersection of the hyperbola and line1 (candidate point1). Or, the end point may coincide with the intersection of the hyperbola and line2 (candidate point2). The appropriate choice of the two candidate points is the one closest along the hyperbola to the known end point. Note that the same reasoning would also apply to a branch segment that is an ellipse or a parabola. This geometric situation is illustrated in Figure 3-2.

The choice is made between candidate point1 and candidate point2 by determining which point is closer to the previous branch segment end point along the conic curve. Conveniently, there is a simple computational test to determine the appropriate point. If candidate point1 is within the sector of arc2, point1 is the appropriate choice. Similarly, if candidate point2 is within the sector of arc1, point2 is the appropriate choice. The case that both of these conditions are true is zero-measure. Furthermore, in that case candidate point1 and candidate point2 are coincident.

Once the end points of the segments have been determined, it is possible to characterize the segment between the end points. The branch segment is known to be a conic section. It is possible to determine the type of the conic section (hyperbola, ellipse, or parabola) by considering the relationship of the associated contour arcs and their respective curvatures. The foci of the conic section are coincident with the centers of the contour arcs. Because the end points of the branch segment lie on the conic section, they provide the remaining information necessary to construct the segment analytically.

Each branch of the skeleton is computed in a piecewise fashion as described above. Each segment of the branch corresponds to two arcs on the curve; the tangent points associated with each point in the branch segment lie on these two arcs. Two neighboring segments always share one arc; the other arcs associated with the two neighboring segments are neighbors on the contour. In the example shown in Figure 3-2, the segment of interest is associated with arc1 and arc2. The neighbor of this segment is associated with arc1 and the neighbor of arc2.

In effect, as the branch is traversed during computation, the tangent points on the contour are implicitly traversed as well. The tangent points associated with each point on a branch segment are easily computed. One of the tangent points is simply

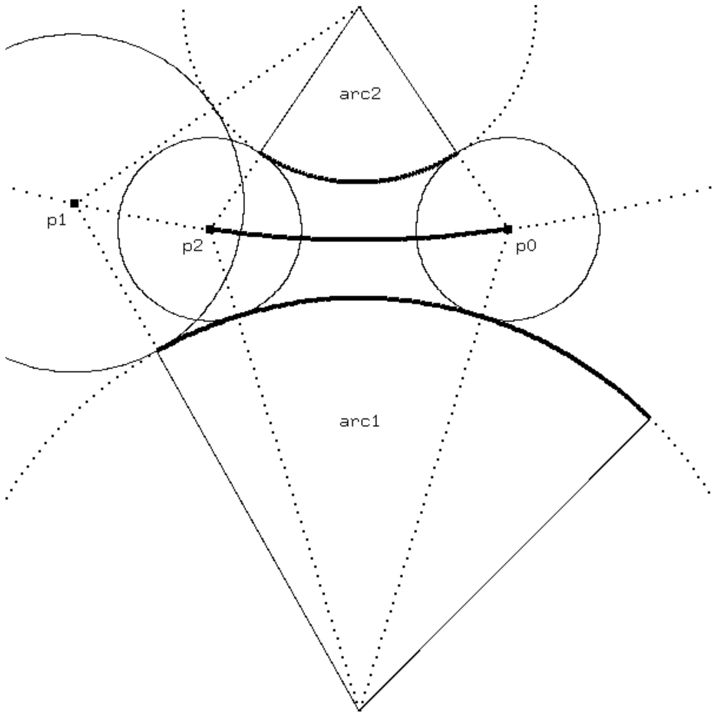


Figure 3-2: Computation of a branch segment. The branch segment is the hyperbolic curve connecting point p0 to point p2. The hyperbola is defined by the property that each point is equidistant from arc1 and arc2 (two arcs in the contour representation). The candidate end point, p1, is the intersection of the end radius of arc1 and the hyperbola. Similarly, the candidate end point, p2, is the intersection of the end radius of arc1 and the hyperbola. The point, p2, is chosen because it is within the sector of arc1 - the point, p1, is not in the sector of arc2. The next segment that would be computed extends beyond point, p2, and is determined by arc1 and the neighbor of arc2.

the projection of the branch point onto one of the arcs associated with the segment. The other tangent point is the projection of the branch point onto the other arc.

Each point on the contour is associated with exactly one point on the skeleton. It is not possible for two distinct skeleton points to have the same tangent point. This property of the skeleton is useful for determining the location of node points on the skeleton, as we shall see.

The skeleton computation begins by determining starting points for candidate skeleton branches. Because it is known that branches terminate into the bounding contour at positive maxima of curvature, these locations are chosen for the starting points. As these candidate branches are extended, the locations of intersections of the branches are found. At the intersections of two branches, a candidate node is formed and an additional candidate branch is created that emanates from the node. During the computation, some of the candidate branches are eliminated when it is determined that no branch exists at its location.

At each positive maximum of curvature on the bounding contour, a candidate skeleton branch is created. By convention, the initial branch segment is the bisecting radius of the arc associated with the maximum of curvature. Strictly speaking, such a segment is not part of the medial axis skeleton as defined mathematically. However, these segments are included by convention because doing so yields more intuitively pleasing results.

Each segment is extended in a piecewise fashion as described above. As the computation proceeds, the algorithm must determine the locations where the branches intersect to form nodes. Each time a branch is extended, the algorithm determines if the branch is overextended relative to another branch. In addition, the algorithm must determine if the other branches are overextended relative to the branch of interest. Ultimately, the algorithm must determine the locations of intersections of branches, that is, the nodes of the skeleton. The following set of conventions achieve these goals.

After a branch has been extended by a single segment, the algorithm determines what interaction, if any, occurs between the branch and the other branches. Conceptually, the algorithm determines if the tangent points of the branch have crossed any of the tangent points associated with any other branch. In practice, the algorithm considers only those branches that have arcs of the contour in common with the branch of interest. More specifically, the algorithm only considers branches whose tail segments (i.e. end segments) have an arc in common with the tail segment of the branch of interest.

If the tail segments of two branches have a contour arc in common, the systematic application of a simple test determines the interaction between the branches. The test determines if either or both of the branches have been extended beyond the intersection between branches. Furthermore, these tests are used to find the node point which is located at the intersection of two segments. This test, described below, is illustrated in Figure 3-3.

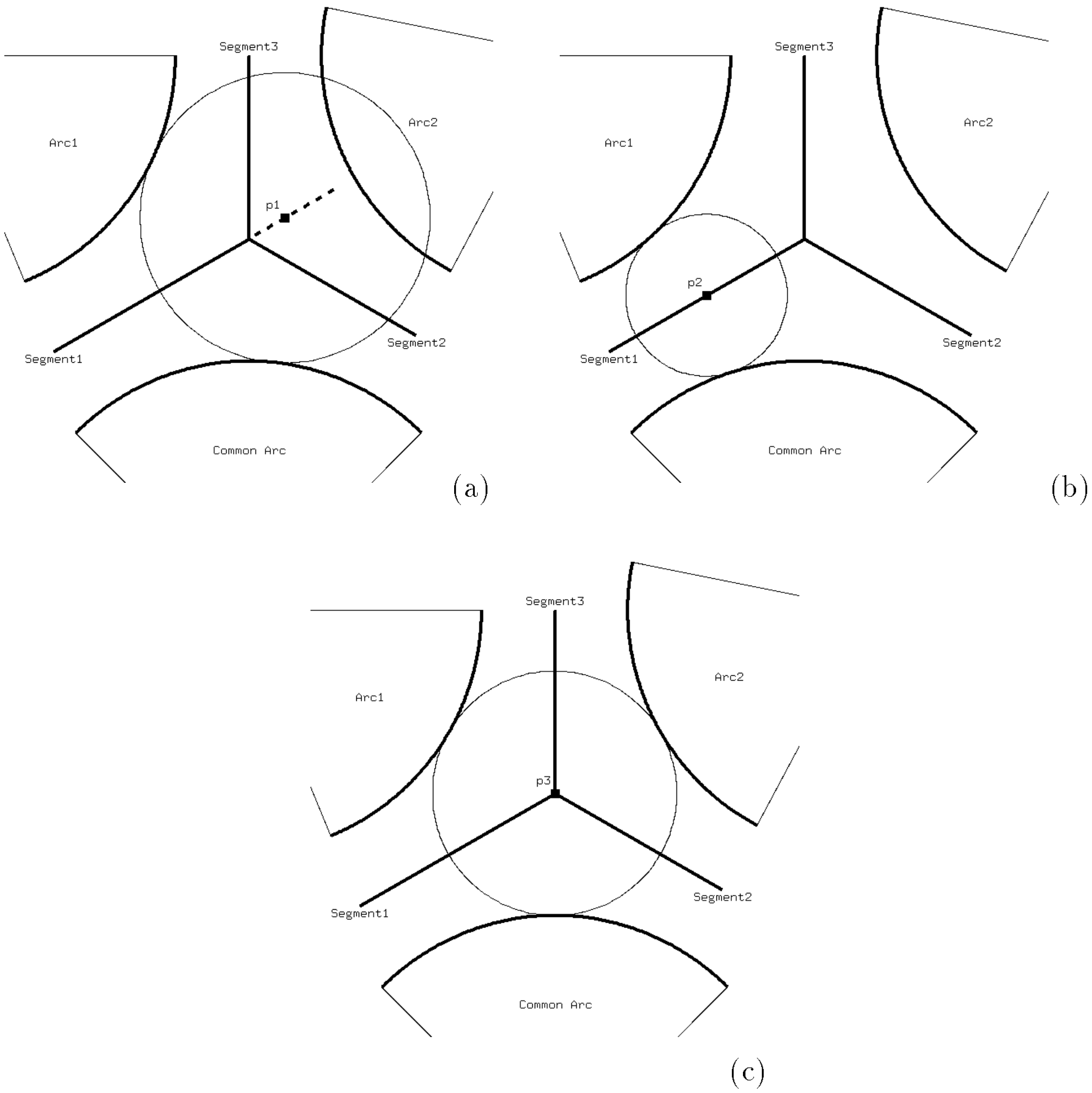


Figure 3-3: Test for determining the location of node points. In each figure, three segments that intersect at a node are depicted. The contour arcs associated with these segments are also depicted. Segment1 is equidistant from the common arc and arc1; segment2 is equidistant from the common arc and arc2; and segment3 is equidistant from arc1 and arc2. Conceptually, segment1 and segment2 intersect to form a node; segment3 emanates from this node. In (a) point p_2 is beyond the intersection of segment1 and segment2: the interior circle is tangent to the common arc and the interior circle intersects arc2. In (b), point p_1 on segment1: the interior circle is tangent to the common arc and arc1, and the interior circle does not intersect arc2. In (c), point p_3 is the intersection between segment1 and segment2: the interior circle is tangent to all three contour arcs.

By convention, the arc associated with both of the tail segments is called the common arc. We arbitrarily refer to one of the branches as branch1 and the other as branch2. Similarly, segment1 and segment2 are the current tail segments of the respective branches. Arc1 and arc2 are the arcs associated with segment1 and segment2 that are not the common arc.

The intersection test determines if an arbitrary point on candidate segment1 is beyond the intersection of segment1 and segment2. The interior circle associated with the point is constructed. By definition the interior circle is tangent to arc1 and the common arc; the center is located at the point of interest on segment1. If the distance from the point to arc2 is greater than the radius of the interior circle, the point of interest is beyond the intersection point. Conversely, if the distance from the point of interest to arc2 is less than the radius of the interior circle, the point is not beyond the intersection of segment1 and segment2. Of course, if the distance from the point of interest to arc2 is equal to the radius of the interior circle, the point is the intersection point.

The intersection test is illustrated in Figure 3-3. In Figure 3-3a, point p_2 is beyond the intersection because the interior circle intersects arc2. In Figure 3-3b, point p_1 is not beyond the intersection because the interior circle does not contact arc2. In Figure 3-3c, point p_3 is the intersection of the two segments; the interior circle is tangent to all three contour arcs.

This test may be used to determine if two tail segments with a common arc intersect. The test is applied to both endpoints for each segment. The segments intersect if and only if the intersection test provides opposite answers for each endpoint of both segments. That is, one endpoint of segment1 is beyond the intersection and the other endpoint is not. Similarly, one endpoint of segment2 is beyond the intersection and the other is not.

When two segments intersect, the node point may be found using the intersection test recursively. Initially, the intersection is known to be between the two original endpoints of segment1. We arbitrarily call these points the upper and lower bound points of the node. The intersection test is applied to a point midway between the bound points. If the midpoint is beyond the intersection, the midpoint becomes the new upper bound. Conversely, if the midpoint is not beyond the intersection, the midpoint becomes the new lower bound. This process is repeated until bound points converge to the node.

It is also possible during the computation that the entire tail segment of a particular branch has been extended beyond the intersection of the branch (branch1) with another branch (branch2). Again, the intersection test is used to distinguish this situation. The test is applied to both endpoints of the tail segment of branch1. If the test determines that both endpoints are beyond the intersection, the entire tail segment is beyond the intersection. In that case, the tail segment is removed from the branch representation.

Note that extending a particular branch is a local operation. It is not necessary to consider the entire bounding contour to perform the computation. In fact, only two arcs of the contour are required for each step. This suggests that the branches could be computed independently, in parallel. Of course, if a branch is extended such that its tail segment has an arc in common with another tail segment, the branches must interact in the manner described above. That is, they must determine if either of the branches is overextended and if the branches intersect at a node.

In our discussion, we have tacitly assumed that the region is bounded by a single simply connected curve. That is, we have assumed that there are no holes in the region. It is straightforward to generalize the algorithm to handle regions with holes. To do so it is necessary to find initial branches such that each segment has one contour arc on an interior curve. (An interior curve is the bounding curve of a hole.) The algorithm extends these initial branches to find nodes similar to the extension of branches described above.

The initial branches associated with the interior curves are found in the following manner: The point on the interior curve that is closest to the bounding curve is determined. Simultaneously, the point on the bounding contour that is closest to the interior curve is found. The midpoint of these points lies on the initial candidate segment for the initial branch. An interior circle is constructed such that the center is the midpoint and the radius is the distance between the midpoint and either of the contour points. The midpoint lies on the skeleton if and only if this interior circle does not intersect another of the interior curves.

In the case that the interior circle does intersect another interior curve, an alternate starting point must be found. The alternate starting point is determined in the same manner described above, except that the closest points between the two interior curves are found. The midpoint between these two points is the new candidate skeleton point. The new candidate point lies on the skeleton if and only if no other interior curve intersects the interior circle. In the event that the interior circle does intersect another interior curve, the process is repeated until an appropriate skeleton point is found. This procedure is guaranteed to provide a skeleton point that is associated with each of the holes.

Once a skeleton point has been found for each hole, a branch segment is constructed that extends in both directions from each of the initial skeleton points. This segment serves as the starting segment for an initial branch. Each of these branches is extended in both directions to find the appropriate nodes with other branches. Figure 3-4 illustrates the initialization of a skeleton corresponding to a region with holes.

Given the piecewise conic description of the skeleton, it is possible to reconstruct the bounding contour exactly. For each segment, it is possible to reconstruct the two contour arcs associated with the segment. If the segment is hyperbolic or elliptic, the centers of the arcs are determined by computing the locations of the foci of the

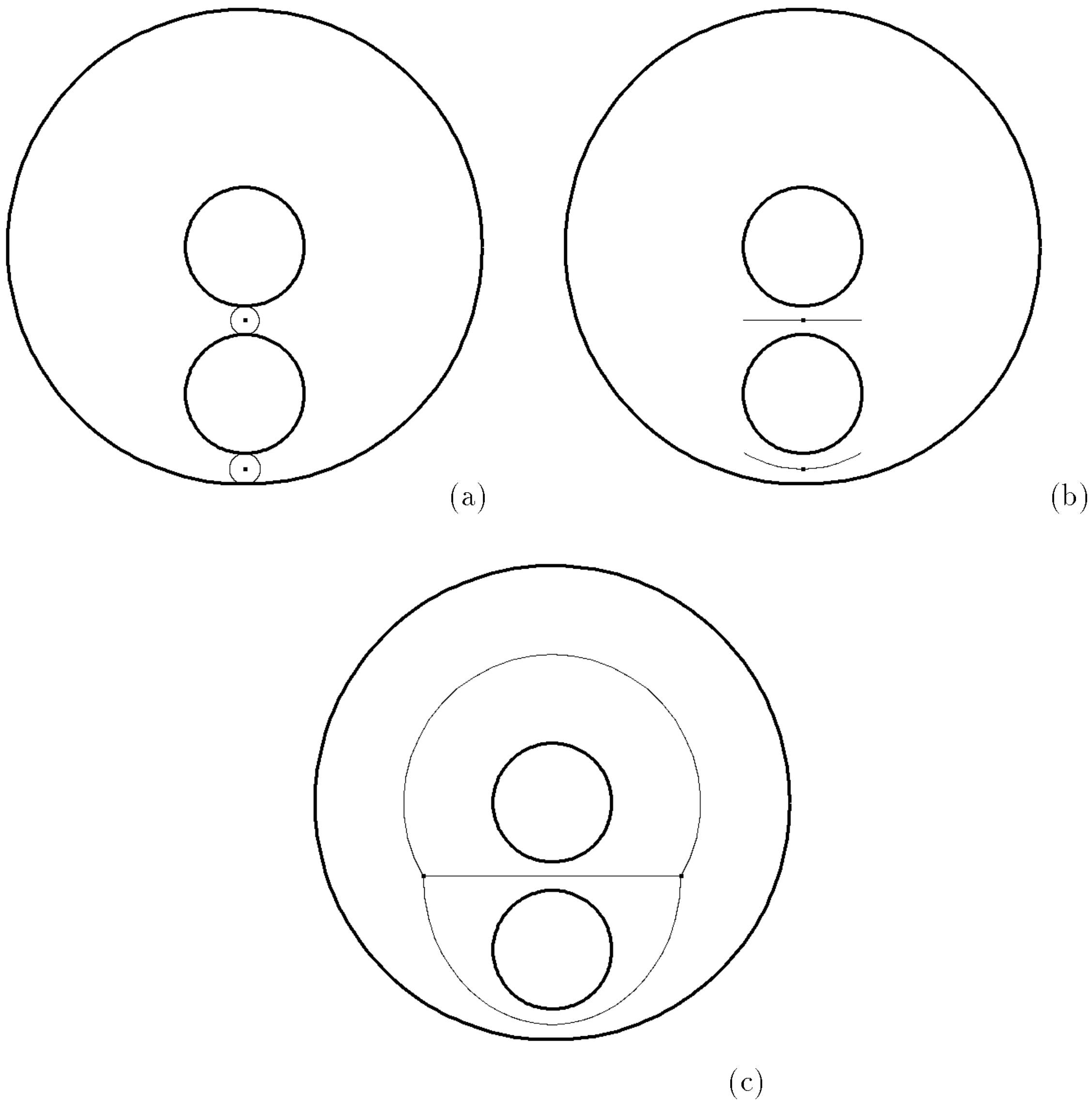


Figure 3-4: Skeleton for a region with holes. The computation of a skeleton for a region with two holes is shown. In (a), the initial skeleton points associated with the interior curves are depicted along with their respective interior circles. In (b), the initial segments are extended from the initial points. These segments serve to initialize the skeleton branches for the computation. In (c), the skeleton associated with the region is shown.

hyperbola or ellipse. If the segment is parabolic, the focus of the parabola is the center point of one arc; the other arc is a straight line segment. The radii of the two contour arcs may be determined by considering the radii of any two interior circles along the conic segment. The endpoints of the contour arcs are determined by projecting the endpoints of the segment onto each arc.

The medial axis skeleton may be computed directly from the analytical contour representation. The contour representation leads naturally to the analytic representation of the skeleton. Each branch of the skeleton is piecewise elliptic, hyperbolic, or parabolic. The bounding contour may be reconstructed exactly from the skeleton.

3.4 The Medial Axis Skeleton Complexity Scale-Space

In Chapter 2, we consider a complexity scale-space for contours. In this section, we extend the concept to define a complexity scale-space for the medial axis skeleton. We consider the computation of the skeleton scale-space from the contour scale-space.

A scale-space is a set of descriptions that differ in their level of detail. At coarse scales the descriptions are relatively simple and, presumably, contain only the most important aspects of the description. At fine scales, the descriptions are relatively complicated and contain the details.

A scale-space representation is desirable because it provides alternative descriptions for subsequent processing. If a higher level process requires accuracy and dense information, a fine scale is appropriate. However, if accuracy is not as critical and sparse information is sufficient, a coarse scale is appropriate. In the latter case, the computational burden is often significantly reduced because the algorithm is required to process a smaller quantity of data.

The construction of a scale-space requires a tradeoff between the accuracy and the level of detail in the description. At fine scales, the tradeoff is skewed toward the accuracy of the description. At coarse scales the tradeoff is skewed toward the simplicity of the description.

The measure of this tradeoff is typically a smoothing parameter such as the spatial width of a Gaussian filter applied to the data (see, for example, [80]). In such a case, an increased spatial width of the filter reduces some of the existing detail. The description is simplified, but the ability to localize the remaining components of the description (the accuracy) is reduced.

In this paper, we propose an novel method of quantifying the accuracy versus simplicity tradeoff. The tradeoff yields a set of descriptions with varying complexity measures, as defined in Section 3.2. Each description is chosen such that it is as close as possible (in the square-error sense) to the data under the constraint that it has a particular complexity measure.

In the case of a contour, we assume that a set of data points along the contour has been provided. A set of contours is constructed such that each contour has a different complexity. That is, each contour has a different number of extrema of curvature. Each of the contours is chosen such that it minimizes the square-error between data points and the contour under the constraint that the complexity measure is equal to a particular value.

In the case of the medial axis skeleton, we also assume that a set of data points along the bounding contour has been provided. Again, a set of skeletons is constructed such that the skeletons have different complexity measures. The bounding contour is chosen such that the square-error between the data and the contour is minimized under the constraint that the associated skeleton possesses the appropriate complexity measure. As we shall see, the contour scale-space is very similar to the skeleton scale-space.

Now, consider the computation of the contour scale-space. If the contour is constrained to pass through each data point exactly, there is a particular minimum complexity measure, M_0. It is not possible to construct a curve that passes through every data point and has a complexity measure smaller than M_0. If the constraint is relaxed such that the curve must pass within some tolerance, δ_1, of each data point, another minimum complexity measure, M_1, is obtained. Of course, $M_1 \leq M_0$. Therefore, as the tolerance, δ, increases, the associated minimum complexity measure, M, decreases. Thus, the tolerance, δ, acts as a scale parameter for the complexity scale-space.

For any tolerance, δ_i, there are infinitely many contours that meet the tolerance requirement and possess the minimum complexity measure, M_i. It is desirable to choose the curve that minimizes the square-error between the data and the contour from the class of minimum complexity curves. This suggests a two-stage algorithm for determining the desired minimum complexity/least square-error curve.

In the first stage, an instance of the minimum complexity contour is computed under the constraint that the curve passes within δ_i of each data point. The curve found in the first stage is used as the starting point for the second computational stage. The output of the first computational stage is illustrated in Figure 3-5 with the silhouette of an airplane as test case.

In the second computational stage, the minimum complexity contour is modified such that square-error between curve and the data points is minimized. The modification is performed under the constraint that the complexity measure does not change. The result is the minimum complexity/least square-error curve. The final results for the airplane silhouette are depicted in Figure 3-6. The computation of the minimum complexity/least square-error curve is described in Chapter 2 and more fully in an earlier paper[15].

As the tolerance parameter increases from scale to scale, contours with smaller complexity measures are found. Some of the details in the contour are eliminated in the process; the more coarse representation is simpler. The features that are present

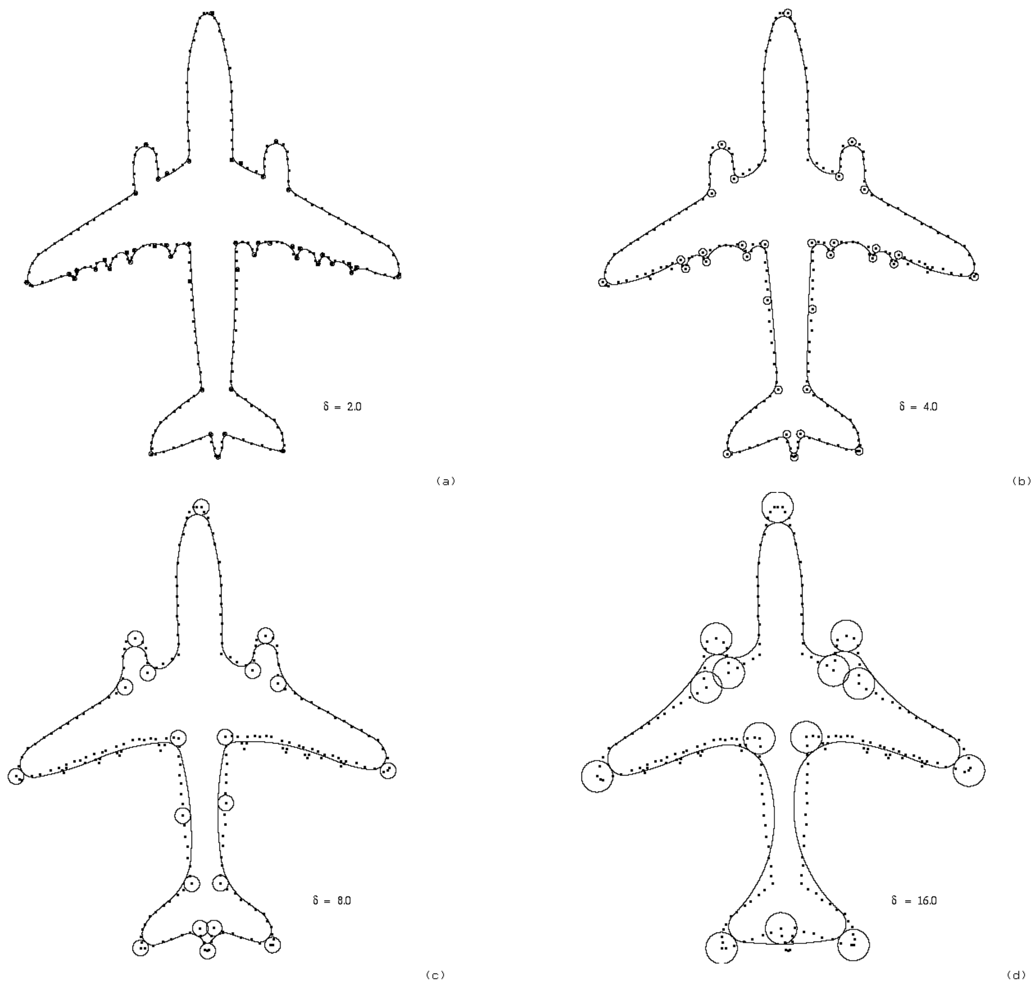


Figure 3-5: Minimum complexity curves for the silhouette of an airplane. Each curve is an instance of a minimum complexity curve for a particular tolerance value. The circles that are tangent to the curve are centered about a particular critical data point. The radius of each of these circles is equal to δ, the scale parameter. The scale parameter for each case is (a) $\delta = 2.0$, (b) $\delta = 4.0$, (c) $\delta = 8.0$, and (d) $\delta = 16.0$.

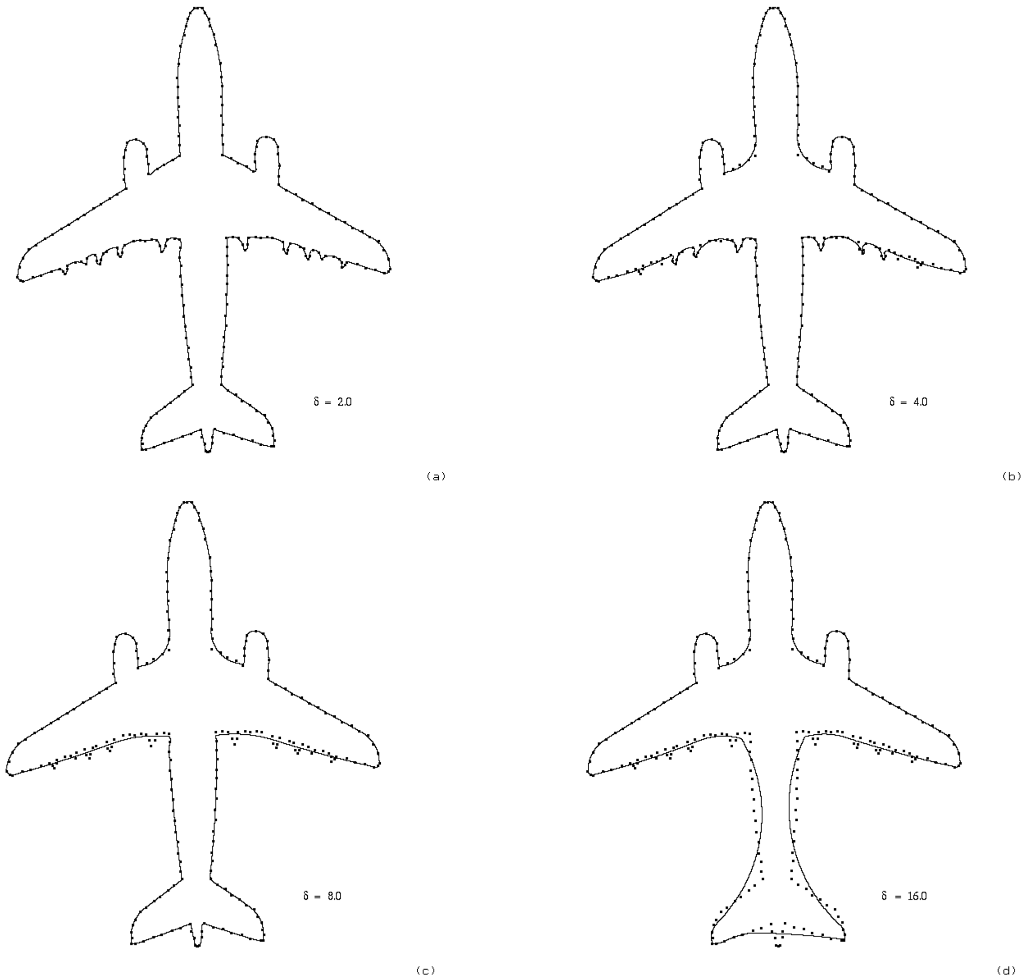


Figure 3-6: Minimum complexity/least-square-error curves for the silhouette of an airplane. In each case the least-square error curve that has the complexity determined from the first stage of the computation is shown. The scale parameter for each case is (a) $\delta = 2.0$, (b) $\delta = 4.0$, (c) $\delta = 8.0$, and (d) $\delta = 16.0$.

in the representation are depicted as accurately as possible. However, the square-error is guaranteed to increase because the contour is unable to account for all of the detail in the data. Thus, as the contour becomes less complex, the accuracy of the representation decreases.

In the case of the airplane silhouette, the position of the tip of each wing is accurately depicted across the entire scale-space. In contrast, at coarse scales, the protrusions on the back of the wings disappear because they are smaller than the scale-parameter. The representation is simpler, because only the gross structure of the wing is depicted. However, the error between the data and the contour is significantly greater in the proximity of the protrusions. This is an example of the tradeoff between simplicity and accuracy.

It would be reasonable to base the skeleton scale-space directly on the contour scale-space. One option is to compute the contour at multiple complexities. For each contour, the corresponding skeleton is computed. As the complexity of the contour decreases, the complexity of the skeleton is guaranteed to decrease. This yields a reasonable scale-space for the medial axis skeleton.

However, there is at least one case where the above definition of the skeleton scale-space leads to a counterintuitive result. This definition does not penalize the magnitude of the curvature; only the number of extrema of curvature is considered. Thus, the algorithm prefers a curve with relatively large curvature if such a choice reduces the square-error, even if the reduction is small. In some cases, this results in an additional skeleton branch that terminates into the curvature maximum. This phenomenon is illustrated in Figure 3-7.

Fortunately, a minor change in the contour smoothing criteria eliminates this effect. The first stage of the contour smoothing algorithm is identical; the curve is modified to minimize the number of curvature extrema. In the second stage, an additional constraint is placed on the computation. No deformations of the curve are allowed that would introduce an additional branch terminus. The complexity of the skeleton is preserved in the second stage, as well as the complexity of the contour.

The change in the smoothing criteria has only a small effect on the flow of computation. In fact, the computation need only be altered for arcs corresponding to positive maxima of curvature. The computation for all other arcs is identical to that for the original criteria.

At each maximum of curvature that is not associated with a branch termination, it is necessary to ensure that the osculating circle extends to the exterior of the contour. Recall that whenever the osculating circle does not extend to the exterior of the contour, a branch terminus is guaranteed to occur at the maximum. Furthermore, whenever the osculating circle extends to the exterior of the contour, no branch terminus occurs at the maximum.

The most obvious method to avoid the introduction of additional skeleton branches is to disallow deformations of the curve that would create a spurious branch. However, this method places an unnecessary burden on the computation. It would be necessary

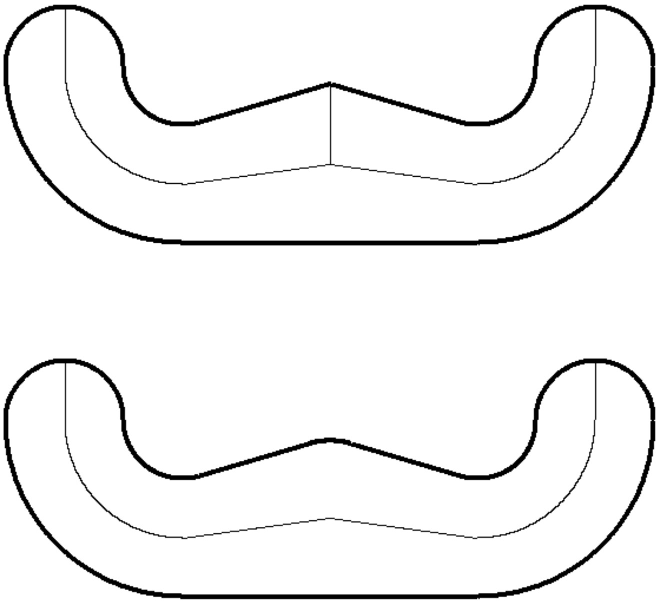


Figure 3-7: Sensitivity of the skeleton at maxima of curvature. Two curves are shown with their respective medial axis skeletons. The two curves are identical except near the maximum of curvature on the top of each curve. The complexity measures of the curves are equal. The top curve has an additional skeleton branch because the magnitude of curvature at the maximum is much greater than that of the bottom curve. The complexity measure of the skeleton is very sensitive to subtle changes in the bounding contour, particularly near positive maxima of curvature.

to test every candidate deformation at each iteration of the smoothing process against this condition.

A more efficient method is to compute the least square-error curve as before, then modify that curve to eliminate any spurious branches that have been introduced. This method requires that the locations of the branch termini are determined after the first stage of the smoothing algorithm. After the second stage of the contour smoothing algorithm, each maximum of curvature is tested to determine if there is a branch present. If a branch has been introduced during the second stage of computation, the curve must be deformed locally to eliminate the spurious branch.

To eliminate a spurious branch, it is necessary to decrease the curvature of the maximum curvature arc. The arc is deformed under the constraint that the neighbors remain fixed, as described in Section 2. The arc is modified until its osculating circle

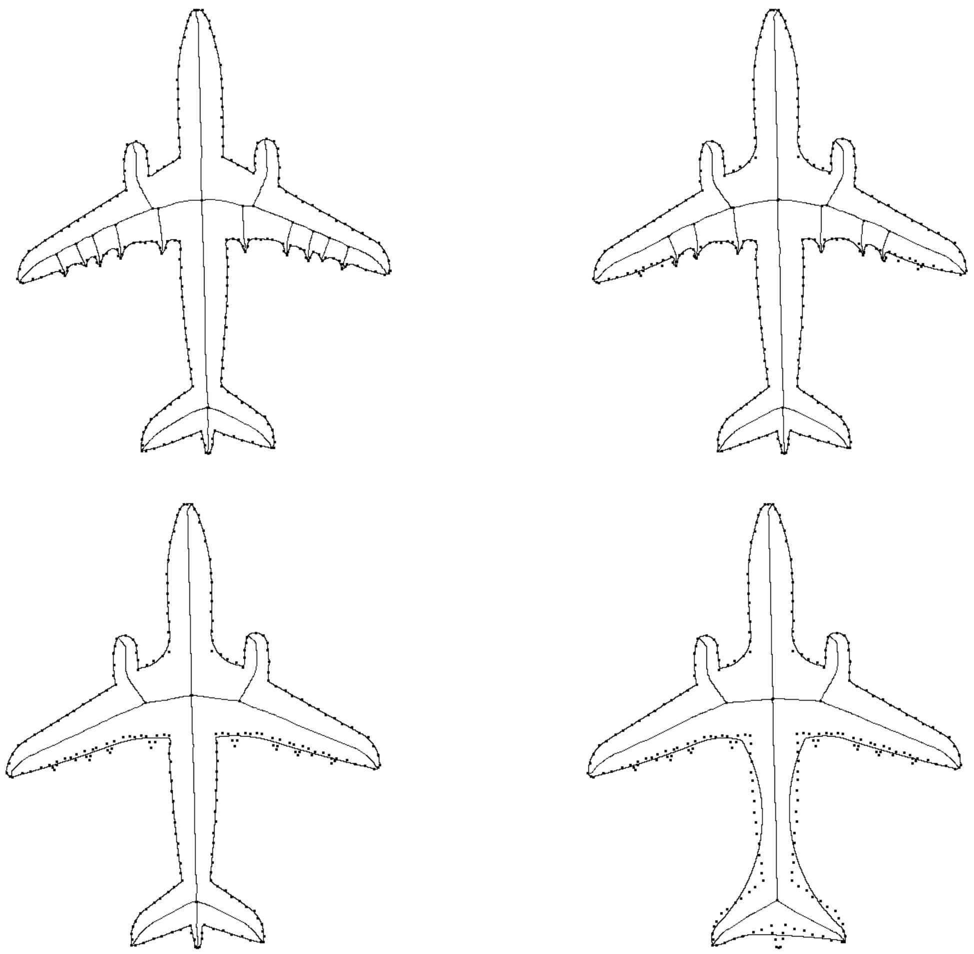


Figure 3-8: The complexity scale-space for the medial axis skeleton. The medial axis skeleton and the corresponding bounding contour are shown at four scales. The scale parameter, δ, is doubled between each scale. Note that the number of branches, as well as the number of features on the contour, decreases as the scale parameter increases. The features that are present in each representation are depicted as accurately as possible in the square-error sense. The small squares near each contour represent the initial data points of the contour.

contacts another portion of the curve; the branch is eliminated. We call the additional point of contact of the osculating circle with the contour the *alternate contact point*.

Once the branch is eliminated, it is desirable to modify the curve locally to reduce the square-error. Such modifications are carried out under the constraint that the osculating circle does not lose contact with the contour at the alternate contact point. This computation may be accomplished locally in the sense that operations need information only about arcs that are near the maximum and the arc associated with the alternate contact point. No other arcs of the contour need be considered.

The result of these computations is a scale-space representation for the medial axis skeleton. At a relatively coarse scale, a skeleton that represents only the gross structure of the interior of the bounding contour is provided. At a relatively fine scale, a skeleton that represents more of the details is provided. The scale-space is illustrated in Figure 3-8 for the silhouette of an airplane.

In the case of the airplane silhouette, the gross structure of the airplane is depicted across the entire scale-space. At the coarse scales, the protrusions of the wings are not present in the skeleton. However, the gross structure of the wings is accurately represented. At the finest scale, the protrusions are present and accurately depicted.

Often, the contours obtained from the contour scale-space are identical with the bounding contours from the skeleton scale-space. However, there are cases where a contour may be modified such that an additional branch appears (or disappears) in the skeleton without changing the number of extrema of curvature in the curve. In these cases, the bounding contour of the skeleton scale-space differs slightly from that of the contour scale-space.

For each type of scale-space, a set of descriptions with varying levels of detail is obtained. The tradeoff between complexity and proximity to the data is quantified in terms of a tolerance about each data point. Each representation is as accurate as possible in the square-error sense under the constraint of minimum complexity. The contour scale-space and the skeleton scale-space are similar; the difference lies in a slight inconsistency of the complexity measures.

3.5 Summary

We have considered a novel approach to computing the medial axis skeleton across a variety of scales. Complexity, as defined by the number of branches of the skeleton, is a natural measure of the level of detail of the skeleton description. The multiple complexity paradigm leads to a true scale-space; the ability to localize a feature is not diminished from one scale to another unless the feature is completely eliminated. The representation for the skeleton is consistent with the corresponding representation of the bounding contour; the mapping between the skeleton and the bounding contour is unique and invertible.

Chapter 4

Discussion

4.1 Comparison of Existing Methods

A variety of approaches have been proposed for interpreting and recognizing contours. Many of these approaches depend on the estimation of properties such as the orientation or curvature of the contour. In the past, a hurdle to such approaches has been the inability of the contour representation paradigms to capture the geometric properties; the estimates are typically extremely noisy. However, the representation presented here provides reasonable estimates for these properties. The improvement in the ability to measure mathematical properties of contours will lead to improvement in the ability to interpret and recognize them.

The results of several existing interpretation paradigms would be improved by use of the analytical representation. For example, the use of codons for classifying contours may be greatly improved by exploiting the advantages of the representation described here. The approach of Hoffman and Richards[36] is to partition the curve at locations of extrema of curvature; the segments of the curve between the extrema are codons. Under the paradigm proposed here, the curvature is represented explicitly in the analytical representation. Furthermore, the complexity scale-space introduced here is natural for their approach. In essence, one may view a reduction of the complexity of the curve to be a reduction in the number of codons allowed to describe the curve.

Another example is the curvature primal sketch of Asada and Brady[3]. The curvature primal sketch is a description of the curve based on primitives delimited by significant curvature changes. If such a set of primitives were desired, they would be extracted more easily from the analytical representation than from the method proposed by Asada and Brady. Furthermore, the complexity scale-space proposed here yields a more well-defined concept of feature or primitive than does the curvature primal sketch.

Similarly, the analytical representation may be used to improve the results obtained by Mokhtarian and Mackworth[57]. In their approach, the shape of the contour is characterized by the positions of inflection points (zero-crossings of curvature) along the contour. Again, if such a representation were desired, the analytical representation would provide a more robust estimate of the zero-crossing position. Furthermore, the analytical representation provides a richer description of curves than does their approach; there is a broad class of curves having no curvature zero-crossings whatsoever (convex curves without straight segments) that are indistinguishable in the Mokhtarian-Mackworth representation.

More recently, Ueda and Suzuki[76] have extended the work of Mokhtarian and Mackworth. They propose a multiple scale contour structure matching algorithm that learns shape prototypes from examplars. The algorithm recognizes unknown contours by comparing the location of the inflection points of the contour to the inflection points of the prototype across the scale-space. In their approach, a multiple scale contour representation is computed by applying Gaussian smoothing to a closed contour with distinct values of sigma.

A critical step in the Ueda-Suzuki process is the computation of the correspondence of inflection points across the scale-space representation of a single contour. That is, as a particular contour is smoothed, neighboring pairs of inflections blend together. From scale to scale, it is necessary to determine which of the inflections have been eliminated, which have been preserved, and the mapping from the inflections in the coarse scale to those of the fine scale. To address this issue, Ueda and Suzuki choose the correspondence that minimizes the sum of the Euclidean distance between corresponding inflections.

Use of the complexity scale-space for contours would simplify the computation of the inflection point correspondence across scales. As the contour smoothing takes place, pairs of inflection points are eliminated by a particular discrete operation on the contour. The inflection points may be labeled prior to smoothing. During the smoothing process, some of the labels are eliminated as inflection pairs are consolidated; the rest maintain their original labels and thereby specify the appropriate correspondence with the finer scale.

In addition, the complexity scale-space could, potentially, facilitate an extension of the Ueda-Suzuki method. The complexity-scale space provides a richer, more useful description of the curvature extrema of the contour than does Gaussian smoothing. Thus, the Ueda-Suzuki matching algorithm could be extended to exploit contour extrema in addition to the inflection points.

The minimum complexity smoothing paradigm has a variety of advantages over existing smoothing methods. For example, the minimum complexity paradigm does not suffer from the shrinkage problem as does Gaussian filtering applied to the data points. The least square-error criterion ensures that there is no bias in the solution.

One disadvantage of the minimum complexity smoothing paradigm is that it is substantially more expensive computationally than other methods. In addition, the minimum complexity paradigm is more complicated to implement than its competitors. However, as the paradigm continues to be developed, it is likely that more efficient and simpler implementations will be found. In the mean time, it remains an open question whether the improved capabilities of the approach outweigh the added computational expense and complexity of implementation.

Recently, Oliensis[59], has proposed a criterion for designing contour-smoothing filters that greatly reduces the shrinkage effect. The Oliensis approach provides an efficient means of computing smooth contours from a set of noisy data points. For many applications, this approach may be sufficient. However, the approach does not provide a complexity scale-space advocated here, nor does it facilitate the computation of the medial axis skeleton.

The analytic contour representation provides the ability to compute the medial axis transform of the region bounded by the contour. It is well-known that the computation of the medial axis transform is extremely sensitive to noise in the contour (see Ballard and Brown[5], for example). Given the ability to reduce the effects of noise on the contour, it is possible to compute the skeleton more reliably.

The contour representation paradigm leads directly to an analytical representation of the skeleton. The skeleton representation consists of piecewise conic sections. The contour and skeleton representations are consist with each other. In fact, it is possible to recover the original contour from the skeleton, if the distance from the skeleton to the contour is given for a finite number of points.

Another advantage of the complexity scale-space is that the medial axis skeleton is consistent with the contour representation. The mapping between the contour and the skeleton is unique and invertible. In contrast, most methods of computing the medial axis skeleton yield a result that is inconsistent with the bounding contour; construction of a curve from the skeleton would lead to a curve that differs from the original bounding contour.

Consistency among representations is desirable because higher level processes may make inferences based upon properties of the contour or the skeleton. If the representations are consistent, such a higher level process is less likely to make incompatible inferences about the data. Such incompatible inferences would lead to degradation of the overall system performance.

4.2 Scale-Space vs. Resolution Space

In Chapters 2 and 3, we introduce two novel scale-spaces based on the complexity measures of contours and the medial skeleton. The tradeoff between accuracy of the representation vs. the simplicity of the description is made explicit. As a result, we draw a distinction between this work and its predecessors.

Most "scale-space" representations would be more accurately described by the term *resolution-space*. Typically, such resolution-spaces are parameterized by the spatial width of some filter, (usually a Gaussian filter, for example [80]). Subjective features are eliminated from the representation as the data is blurred. For example, at a fine scale a particular feature may be represented accurately. At a coarse scale, the feature may be eliminated, as desired. However, at an intermediate scale, the feature may exist with degraded accuracy. There is no advantage to representing particular atomic features with varying degrees of accuracy.

A true scale-space provides descriptions of the data with varying levels of detail. At each scale, all features are represented as accurately as possible. Given that a feature is present at two scales, there is no advantage in reducing the accuracy of localization of the feature from one scale to the next. It is more desirable to retain the accuracy of the representation from one scale to the next, until a feature is eliminated altogether.

The complexity measure yields a true scale-space for contours and medial axis skeleton. At finer scales, greater detail is represented. At coarse scales, only the gross structure of the skeleton is represented. The complexity criterion explicitly guarantees that the representation becomes simpler at coarse scales.

The medial axis skeleton complexity scale-space offers a novel approach to the trade-off of accuracy and simplicity. The tradeoff occurs between the number of subjective features and the square-error between the data and the representation. Whenever a feature is present in the representation, it is depicted as accurately as possible. For example, the location of the tip either wing of the airplane depicted in Figure 3-6 is represented accurately across the entire scale-space. Of course, there is still a reduction in the overall accuracy of the representation at coarse scales because, for example, the protrusions on the back side of each wing are not depicted at the coarse scales.

In general, it is more difficult to obtain a true scale-space representation than a resolution-space. It is non-trivial to obtain a formal tradeoff between the simplicity of a description and the accuracy. Whenever possible, it is desirable obtain a true scale-space representation. Of course, resolution-spaces are useful to the extent that they approximate the desired behavior of a scale-space.

4.3 Minimum Description Length Coding

The complexity scale-space is similar to the minimum description length (MDL) approach at an intuitive level. The idea of providing the simplest possible representation is present in both approaches. However, the formal definition of complexity is different in the two approaches. As a result, there are significant conceptual differences between the complexity scale-space and the MDL paradigm.

The MDL criterion requires that the number of parameters employed by a model to account for the data should be minimized[66]. More precisely, the number of bits required to encode the data is minimized. Thus, an important component of the MDL approach is the choice of an efficient model for the representation. Formally, the MDL approach requires that the representation correspond to an optimal code in the information theoretic sense. A theory for choosing the representation from *a priori* probability distributions is well known (see Leclerc[46], for example).

Unfortunately, the determination of such an optimal code for a general application is difficult in practice. Typically, minimum description length approaches assume a particular form of the representation. The number of parameters employed by the representation is minimized. Interestingly, the MDL theory provides a simple objective measure of the performance of a particular representation. The performance measure is, of course, the average length of the code.

For example, Leclerc[46] applies the MDL technique to the image partitioning problem. Each subregion requires a set of parameters to distinguish the boundary and to specify the behavior of image brightness within the subregion. The additional parameters required for each subregion causes the algorithm to favor a small number of partitions. However, when there are relatively few partitions, it is more difficult to account for the variations from the model within each subregion; a larger number of bits per subregion is required. Conversely, this effect favors a larger number of partitions. There is a particular choice of the partitioning that optimizes these two opposing effects.

In contrast, the complexity scale-space seeks to minimize the number of subjective features in the description. The optimization is indifferent to the number parameters used by the representation. In the case of a contour, the contour may be described by arbitrarily many circular arcs; the complexity depends only on the number of extrema of curvature. Similarly, each branch of a medial axis skeleton may be represented by arbitrarily many branch segments; the complexity is specified by the number of branches. An MDL approach would essentially require minimizing the number of arcs in a contour or the number of segments in a skeleton.

Furthermore, the complexity scale-space does not attempt to provide an efficient code for the representation. Rather, the scale-space seeks to make the most relevant information explicit for higher level processes. In fact, there is a loss of information at the coarse scales. We have argued, above, that this loss of information is desirable for computational purposes.

Another important distinction is that MDL approaches typically do not provide scale-space representations. MDL approaches provide a single representation for each data set. In effect, an optimal scale is chosen implicitly by the MDL criterion. The scale chosen by the MDL criterion is the one requires the fewest number of bits to represent the data.

The paradigm described in Part I of this thesis leads to true scale-spaces for contours and the medial axis skeleton. At first glance, the complexity scale-spaces are very similar to minimum description length approaches; the notion of simplicity of the representation is exploited in each case. However, there are significant philosophical and practical differences between the complexity scale-spaces and MDL approaches.

4.4 Summary of Part I

In Part I of this thesis we consider a framework for attacking a variety of computational geometry problems. We begin by establishing a paradigm for representing contours. The paradigm allows us to represent geometric properties, such as position, orientation, and curvature, explicitly and exactly. The ability to represent these properties leads us to improved computational capabilities.

The representation facilitates smoothing the contour using a novel criterion. The complexity of the contour is minimized under the constraint that the curve passes within a specified tolerance of the data points. Given that complexity, the least-square error curve is obtained. The tolerance acts a smoothing parameter. The smoothing operation leads to a novel scale-space that exploits a trade-off of the complexity of contour versus the accuracy of the description.

From the contour representation, it is possible to compute an analytical representation of the medial axis skeleton. Unlike other approaches to computation of the skeleton, the resulting skeleton is consistent with the bounding contour. In fact, the contour may be reconstructed exactly from the skeleton representation, if the distance between the skeleton and the contour is provided at a finite number of points.

The complexity scale-space for contours is extended to the medial axis skeleton. The contour is chosen such that the complexity of its associated skeleton is minimized. Again, the tolerance acts as a scale-parameter.

In Part II of this thesis, we consider the application of the computational paradigm to feature extraction, feature tracking, and the estimation of egomotion. The abilities we develop in Part I facilitate the computation of a robust set of features. The improvement in the ability to compute and represent geometric properties of the features makes the feature tracking problem more manageable. As a direct result, we are able to improve our estimates of egomotion and depth as the image sequence evolves in time. Thus, improvement of abilities at the most primitive level leads to improved capabilities in the higher level processing.

Part II

A Feature Extraction Paradigm

Chapter 5

Simple Region Features

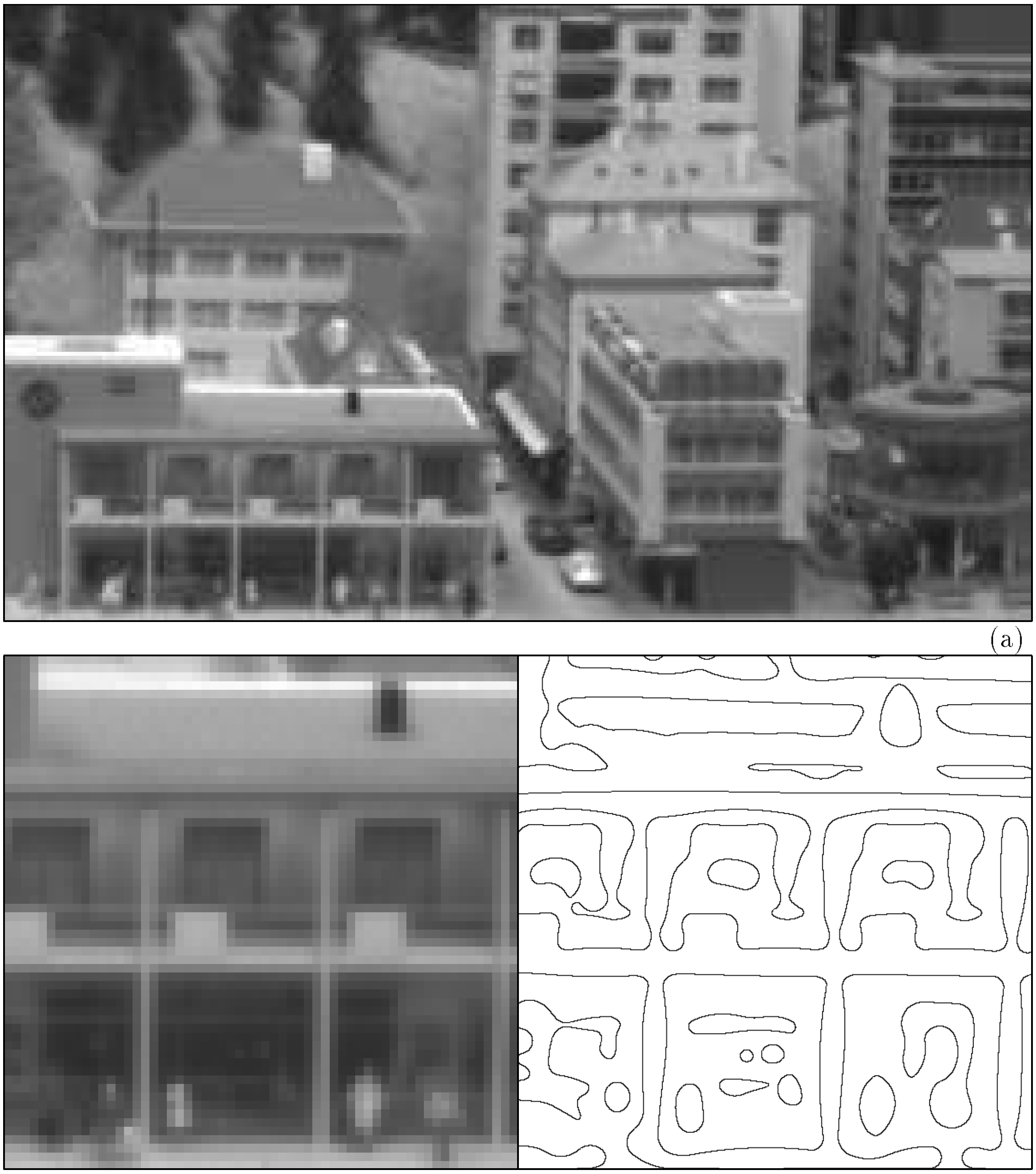
5.1 Introduction

In Part II of this thesis, we extend the geometric abstractions developed in Part I. We apply the smoothing paradigm of Chapter 2 and the skeleton computation of Chapter 3 to the Laplacian of Gaussian zero-crossings at multiple resolutions. As a direct result of the improved processing capabilities, we are able to compute a novel set of features from the zero-crossings. The features, called *simple regions* or *simple region features*, are the basis for an early vision paradigm introduced in subsequent chapters. We define the paradigm and demonstrate its viability by applying it to the passive navigation problem.

Since the work of Marr and Hildreth[53], zero-crossings of the Laplacian of Gaussian filter have often been interpreted as the locations of brightness discontinuities or edges in the image. The Laplacian is a second order differential operator whose zero-crossings approximate the location of the maxima of the brightness gradient[7]. The Gaussian filter is combined with the Laplacian to make the derivative operation more well-conditioned[31], [72]. The resolution of the edge detection operator is controlled by the spatial width of the Gaussian filter.

In addition, the Laplacian of Gaussian operator may be interpreted as a matched filter that is tuned to objects of a particular width. The size of objects for which the filter is tuned is controlled by the spatial width of the Gaussian, σ. The filter responds maximally at the center of an object whose width is approximately 2σ. (The width of the main lobe of the filter is 2σ.)

Taken together, these interpretations suggest the (naive) idea that the Laplacian of Gaussian filter may be used to extract features of a particular size from the image. The filter responds maximally in the center of objects with width near some optimal value. The zero-crossings mark the location the edges (i.e. the boundary) of the object.


(a)

(b) (c)

Figure 5-1: Zero-Crossings of Model Image. In (a), an image of a model town is depicted. In (b), a portion of the image is extracted. The sub-image corresponds to the front of the building in the lower left of the full image. In (c), zero-crossings for the subimage are shown.

Of course, the regions associated with nearby objects tend to merge or blend together. This results from the blurring introduced by the Gaussian filter. Rather than segmenting the image as desired, the zero-crossing regions tend to meander over the image. A particular region typically covers multiple objects in the image rather than extracting a single object or a single subjective feature.

However, the salient parts of the zero-crossing regions often correspond to subjective features in the image. Consider, for example the image and its zero-crossings depicted in Figure 5-1. Overall, the zero-crossing regions meander over much of the image. However, much of the structure present in the image also appears in the regions. For example, the pillars in the front of the building are identifiable in the zero-crossings.

If it were possible to decompose the zero-crossing regions into their salient parts, the subregions would be candidates for use as atomic tokens or features in subsequent processing. It would be possible, for example, to recognize objects based on the configuration of the groups of the subregions that make up the object. It would also be possible to track the subregions across sequences of images and thereby estimate the egomotion of the camera and the structure of the environment.

In this chapter, we define a particular decomposition of a region into salient parts. The decomposition is robust against small perturbations in the bounding contour. As we shall see, continuous deformation of the bounding contour leads to continuous deformation of the subregions except at critical occurrences. At the critical occurrences, the subregions split or merge in a predictable manner.

The decomposition is applied to regions bounded by the zero-crossings of a Laplacian of Gaussian filtered image. There are two complementary sets of regions. The positive regions correspond to regions of the image where the output of the filter is positive. Likewise, the negative regions correspond to regions of negative output of the filter.

Features are obtained at multiple resolutions by applying the decomposition to the outputs of a number of Laplacian of Gaussian filters. The filters differ by their spatial width or bandwidth. At a relatively narrow spatial width (wide bandwidth) of the filter, the subregions typically correspond to elements of detail in the image. Conversely, at a relatively wide spatial width, the subregions typically correspond to the gross structural elements in the image.

The primary advantage of this approach is that the features correspond to salient regions in the image rather than points. As a result, the features possess a more abstract set of attributes. Each region has a centroid (position), area, and orientation. The shape of each region may also be described. And, as we shall see, the topological relationships among groups of features is specified explicitly.

The availability of abstract information pertaining to the features yields tremendous advantages. In particular, we shall exploit the availability of the area, shape, and the topological information in Chapter 6 when we consider the correspondence

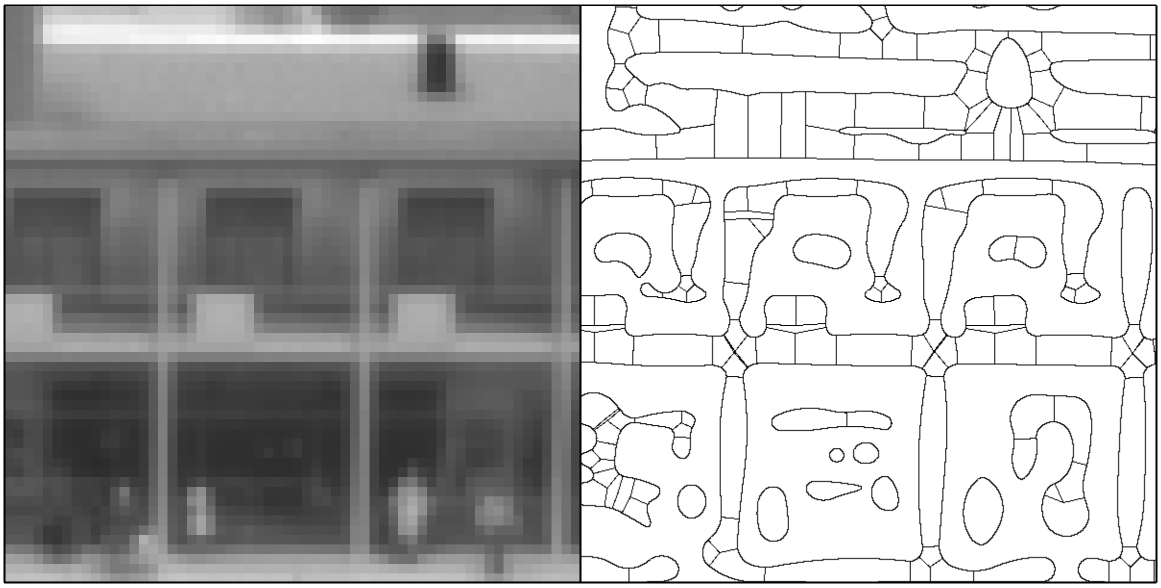


Figure 5-2: Simple Region Features of Model Image. In (a), the subimage of Figure 5-1b is depicted. In (b), the positive simple region features for the image are shown.

problem over a sequence of images. In addition, we argue in Chapter 8 that the additional attributes of the features are advantageous in a wide variety of contexts.

In contrast, point features, such as Canny edges[13], have only a position attribute (and a rough estimate of the orientation). One might argue that more abstract features could be obtained by using point features as primitives. Pairs of such points may be coupled to form line segments; this yields an orientation attribute, as well as position. Groups of neighboring line segments may be joined to form curves, thereby adding a curvature attribute. However, such groups are often unstable and sparse.

Before defining the feature extraction process in more detail, it is instructive to consider an example of features that we will ultimately extract. In Figure 5-2, an image and its corresponding positive simple regions at the finest resolution are shown. Note that much of the structure present in the image is represented explicitly by the simple regions. For example, the pillars of the lower level of the building correspond to vertical simple regions. Of course, in areas of the image where the contrast is low, the features are not as distinctive. For example, the pillars of the upper level of the building are not as well-defined. However, as we shall see, even in such areas, the decomposition remains relatively stable as the image sequence evolves in time.

In this chapter, we define simple region features more precisely. We start by considering the appropriate decomposition of regions into their salient parts. Once we have settled on the decomposition, we specify the computation of the features, including relevant geometric properties. Finally, we discuss the overall feature extraction process and general properties of the features.

5.2 Region Decomposition

The crucial step for computing simple region features is the decomposition of a region bounded by zero-crossings into its salient parts. This computation allows us to consider small pieces of the zero-crossing regions independently. The decomposition is based on the medial axis skeleton computation described in Chapter 3 and the analytical contour representation defined in Chapter 2.

Consider the shapes depicted in Figures 5-3a and 5-3b. For each shape, the medial axis skeleton is superimposed in the figure. The shape on the left consists of three subjective parts: the three bars extending from the center. Similarly, the shape on the right consists of three subjective parts: the three round bulges that have merged together

The goal of the region decomposition is to break the region into these perceptually obvious subparts automatically. The medial axis skeleton is the basis for the decomposition. The skeleton is a natural choice because it represents the structure of region. Each branch of the skeleton corresponds to a salient part of the region. However, the medial axis skeleton does not achieve the desired decomposition by itself. It is necessary to extend the definition to break branches of the skeleton at locations where the distance from the skeleton to the contour is a local minimum.

The desired decomposition is achieved for the three bar shape by the medial axis skeleton. The skeleton consists of three branches, one for each bar. In Figure 5-3c, the subregion corresponding to each the three branches are depicted. The desired decomposition is achieved.

The three bulge shape is not successfully decomposed by the medial axis skeleton, alone. The respective skeleton consists of only a single branch. To achieve the desired decomposition, the branch must be broken at the locations of minima in the distance between the branch and the contour. The locations of these minima are depicted in Figure 5-3d. Segmenting the region at these location achieves the desired decomposition.

The region decomposition suggested by Figure 5-3 may be defined functionally in two stages. First, the region is segmented into initial subregions that correspond to the branches of the medial axis skeleton. Each initial subregion is further segmented at the locations of minima in the distance between the skeleton branch and the boundary. An algorithm that computes the decomposition is specified in the next section.

5.3 Computation of the Decomposition

Computing the region decomposition defined above is straightforward given an analytical representation of the bounding contour and the medial axis skeleton described in Chapters 2 and 3. The first step of the process is achieved implicitly with the computation of the skeleton. The second step is achieved by exploiting geometric properties of the contour representation.

Each branch of the skeleton corresponds to an initial subregion of the original region. The initial subregion is defined by the portion of the bounding contour that is associated with the branch and line segments that demark the boundary between neighboring subregions. The line segments separating the initial subregions are defined by the common skeleton node point and the radial projection of the node point onto the bounding contour. In the case of the three bar region depicted in Figure 5-3a, there is a single skeleton node at the center of the region. The line segments defined by the projection of the node point onto the bounding contour act as the boundary between adjacent subregions.

Recall from Section 3.3 that a skeleton node point is equidistant from three points on the bounding contour. As a result, there is a circle of interest whose center is coincident with the node point. The circle of interest is tangent to the bounding contour in three places. The projections of the node point onto the bounding contour are coincident with these tangent points. Therefore, the line segments that act as the boundary between neighboring subregions are radii of the circle centered at the node. These geometric relationships are illustrated in Figure 5-3e.

Each initial subregion of the decomposition is represented by the bounding contour associated with the branch and the line segments demarking the boundary with neighboring subregions. The portion of the bounding contour associated with a particular branch is represented explicitly in the skeleton representation. Recall that each segment of a branch corresponds to two particular circular arcs in the contour representation. The line segments are computed by determining the radial projection from the skeleton node point onto the appropriate contour arcs.

Once the initial decomposition has been computed, it is necessary to divide the initial subregions at locations of minima in the distance between the branch and the contour. When such a minimum occurs, the skeleton point is collinear with its two tangent points on the bounding contour. (Recall from Section 3.3 that the tangent points are the two points on the contour closest to the skeleton point.) The line segment that extends between the tangent points and includes the skeleton point is a diameter of the interior circle of the skeleton point. (Recall also that the center of interior circle is the skeleton point; the interior circle is tangent to the bounding contour at the tangent points; and the interior circle does not contact the bounding contour, except at the tangent points.) Because the interior circle is tangent to each of the two contour arcs, the centers of the contour arcs are also collinear with the

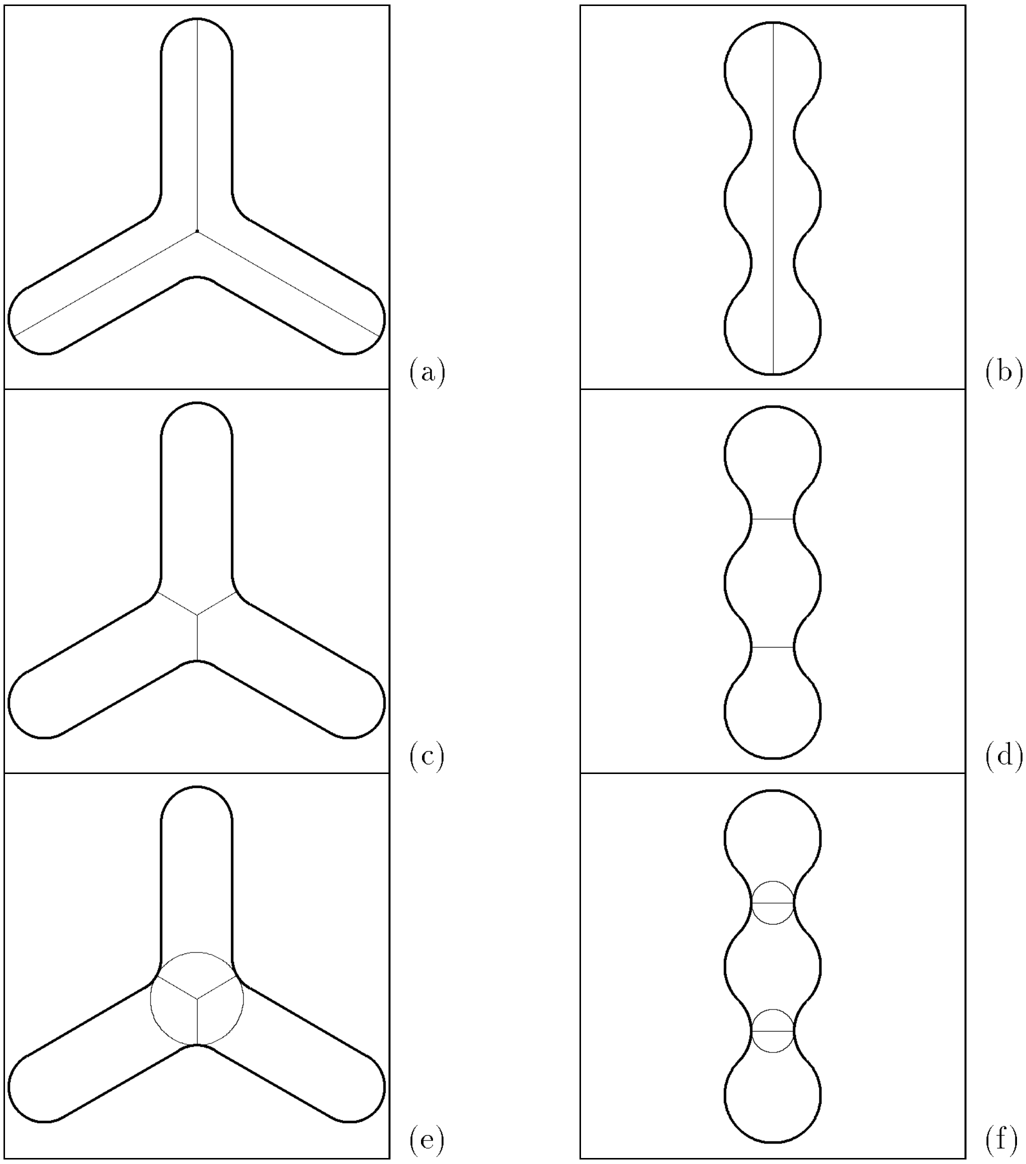
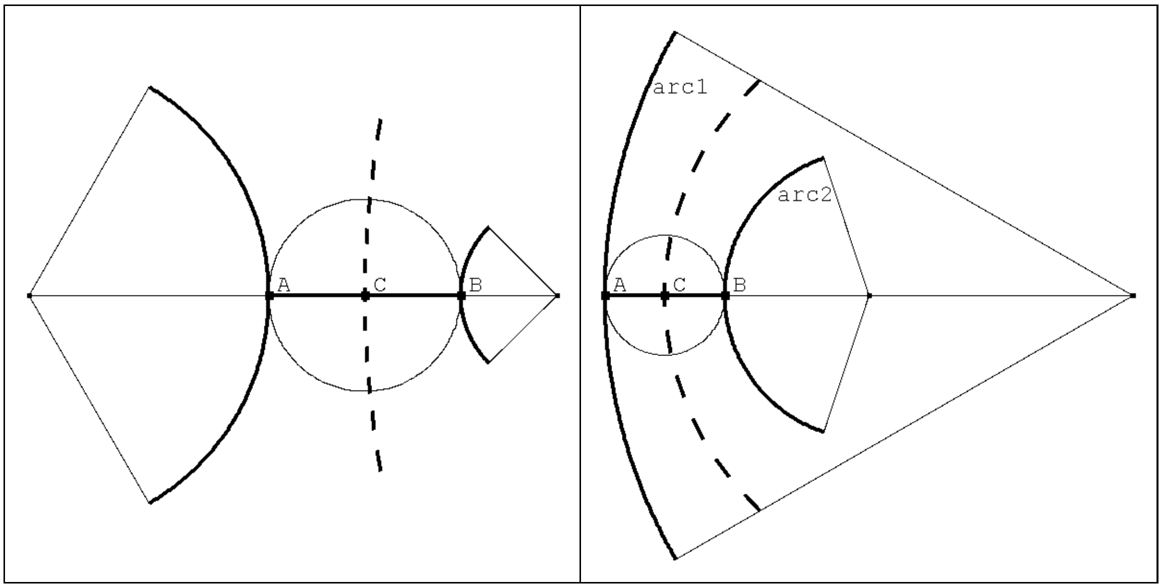


Figure 5-3: Decomposition of Example Regions. In (a) and (b), the bounding contours of two contrived shapes are shown along with their respective medial axis skeletons. In (c), the shape from (a) is segmented into the three regions associated with each of the skeleton branches. In (d), the shape is segmented at the locations of minima in the distance between the contour and the skeleton. In (e), the interior circle associated with the skeleton node point is depicted. The line segments that bound each subregion are radii of this circle. In (f), the interior circles associated with the skeleton points at the boundary are shown. The line segment associated with each boundary is a diameter of the respective interior circle.



(a) (b)

Figure 5-4: Geometric Relationships at Distance Minima. In each figure, a skeleton branch segment and its associated two contour arcs are shown. The branch segment is the dashed line. A minimum in the distance between the contour the branch occurs within the segment. The centers of contour arcs, the skeleton point at the minimum (C), and the tangent points associated with the skeleton point (A and B) are all collinear, as indicated. The line segment AB is a diameter of the interior circle of point C. In (a), both contour arcs have negative curvature. A minimum is known to occur because each arc center is within the sector of the other arc. In (b), arc1 has positive curvature and arc2 has negative curvature. Point A is the radial projection of the center of arc2 onto arc1. Point B is the radial projection of point A onto arc2. A minimum in the distance is known to occur because A is within the sector of arc1 and B is within the sector of arc2.

tangent points and the skeleton point. These geometric relationships are illustrated in Figure 5-3f and Figure 5-4a.

We exploit the collinearity of the skeleton point, its tangent points, and the centers of the contour arcs to compute the locations of the minima in the distance function along the skeleton branch. For each branch segment, a simple test determines if a minimum in the distance function occurs. At least one of the contour arcs must have negative curvature. If both contour arcs have positive curvature, no minimum exists in the branch segment. If both contour arcs have negative curvature, a minimum occurs if and only if the center of each arc is within the sector of the other. The test is a direct result of the collinearity of the five points of interest. The geometric relationships involved in this test are illustrated in Figure 5-4a.

If one of the contour arcs has positive curvature and the other negative, a slightly more complicated algorithm is needed to determine if a minimum is present. Two conditions must be met for a minimum to occur. First, the center of the negative curvature arc must lie in the sector of the positive curvature arc. Equivalently, the radial projection of the center of the negative arc onto the positive arc is within the positive arc sector. We call this projection point A. Second, the radial projection of point A onto the negative curvature arc must lie within the sector of the negative curvature arc. We call this projection point B. The two projection points, A and B, are the tangent points of the interior circle. By construction, the projection points are collinear with the centers of the contour arcs. In addition, the skeleton branch point is the midpoint of A and B (point C). Point C is also guaranteed to be collinear with center points. These relationships are illustrated in Figure 5-4b.

Once the decomposition is computed, a graph representing the adjacency of simple regions is constructed. At each end of a simple region, there may be zero, one, or two neighbors. There are zero neighbors when the skeleton terminates into the contour. There is one neighbor when the boundary is the result of a decomposition at a distance minimum along the skeleton. There are two neighbors when the boundary occurs at a skeleton node. The purpose of the connectivity graph is to allow subsequent processes to determine the neighbors of any simple region efficiently.

5.4 Extraction of Simple Region Features

Now that we have considered the relevant region decomposition, it is desirable to consider the entire process. We begin with a set of Laplacian of Gaussian filters with varying resolutions. At each resolution, we extract and smooth the zero-crossings. The regions bounded by the zero-crossings are decomposed into salient subparts. These subparts are the desired features.

The first step of the feature extraction process is to pass the image through a set of Laplacian of Gaussian filters. Technically, the filters are approximations to the Laplacian of Gaussian. However, for our purposes, the approximation is valid.

The Gaussian filtering is approximated by a sampled Gaussian filter, since the data is acquired in the discrete domain. At the finest resolution, the image data is smoothed with a filter with a standard deviation $\sigma = 1$ in units of the picture cell spacing. The next resolution is obtained by applying a sampled Gaussian filter with standard deviation $\sigma = \sqrt{3}$. This value is chosen because two cascaded Gaussian filters are equivalent to a single Gaussian filter whose variance is the sum of the variances of the individual filters. (Of course, the variance is the standard deviation squared.) The effective total smoothing after the application of the second filter is $\sigma = 2$ ($\sigma^2 = 4$).

The output of the filter is subsampled by a factor of 2×2. The bandwidth of the filter applied prior to subsampling is well within the Nyquist criterion to avoid aliasing. After the subsampling, the data has been effectively smoothed by a Gaussian with $\sigma = 1$ when measured in the subsampled coordinates. Subsequent resolutions are obtained by applying the filtering and subsampling process recursively. That is, the data at one resolution is filtered by a Gaussian with a standard deviation $\sigma = \sqrt{3}$ and subsampled by 2×2 to obtain the smoothed data for the next coarse resolution.

The result is a set of smoothed and subsampled versions of the image data. Such a construct is often called the Gaussian pyramid[12]. The standard deviation at the finest resolution is 1 when measured in units of picture cell spacing. In absolute terms, the effective smoothing varies by a factor of two from resolution to resolution. This choice is consistent with previous work using the Laplacian of Gaussian resolution space (see, for example, Grimson[26]).

The Laplacian component is approximated by applying a 3×3 symmetric filter mask to the data at each resolution. The center value of the mask is $\frac{20}{\sqrt{468}}$, the side values are $\frac{-4}{\sqrt{468}}$, and the corner values are $\frac{-1}{\sqrt{468}}$. The constant $\frac{1}{\sqrt{468}}$ is chosen because it normalizes the energy contained in the operator to unity. The 3×3 mask is a second order difference approximation to the Laplacian operator. The result is a Laplacian pyramid structure that is similar in spirit (though different in detail) to the one proposed by Burt and Adelson [12].

The Laplacian approximation is applied to the data after the subsampling at each resolution. This differs from some approaches in which the Laplacian approximation is applied once to the data at the finest resolution. In the other approaches, smoothing and subsampling is typically applied recursively to the output of the Laplacian approximation. A block diagram of each of the alternative processing scenarios is depicted in Figure 5-5.

We choose to apply the Laplacian operator after subsampling, as shown in Figure 5-5a, because doing so results in a Laplacian pyramid that is self-scaling. The effective filters that are applied to the resolutions of the pyramid have the same frequency characteristics, when considered in subsampled units. As a result, any difference in the spectral properties of the data from resolution to resolution is strictly due the spectral content of the original image data. This statement is not true for

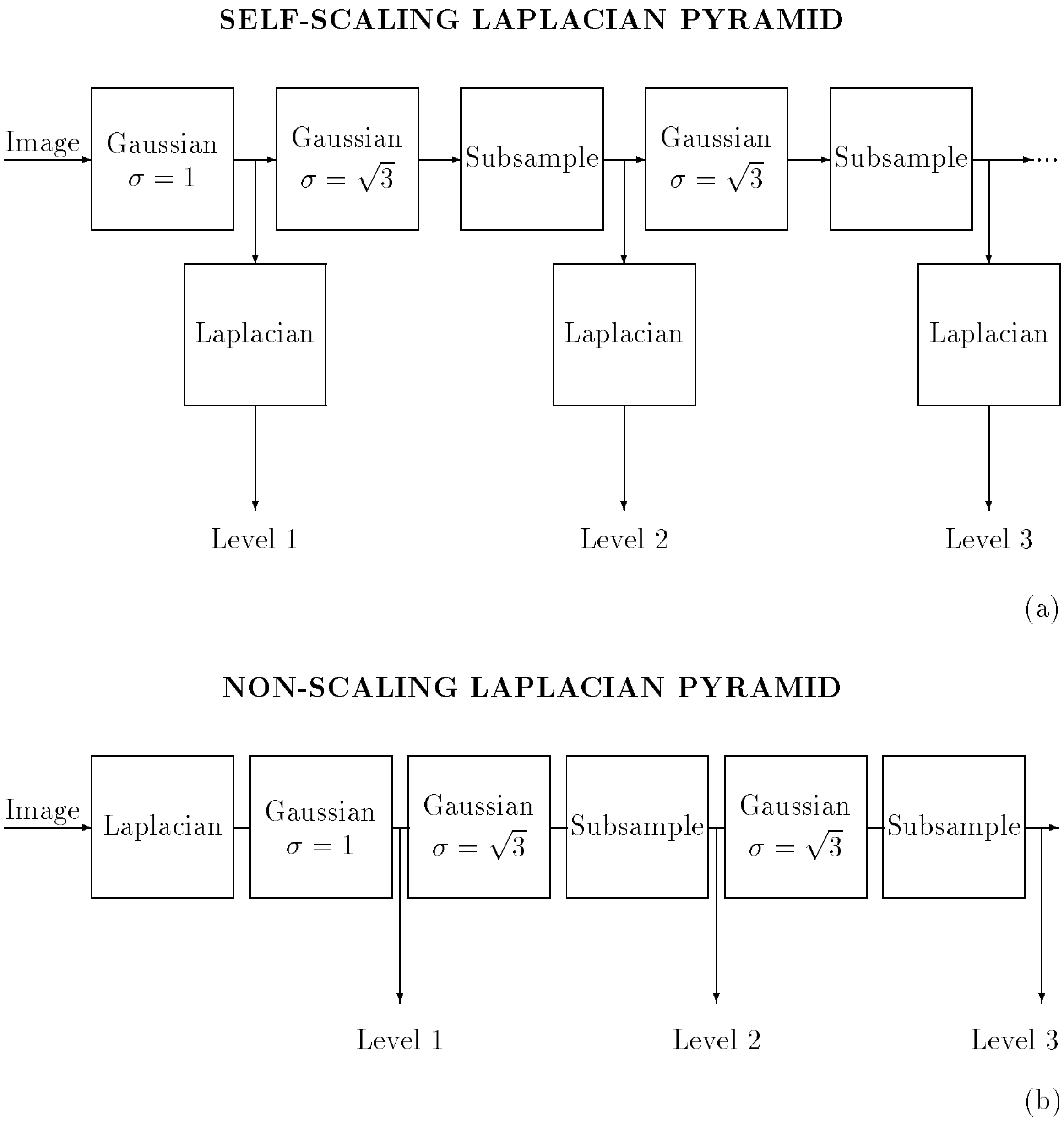


Figure 5-5: Laplacian Pyramid Alternatives. In (a), the self-scaling Laplacian of Gaussian filter bank is shown. The Laplacian approximation is applied to the signal after subsampling at each resolution. This results in identical frequency response of the effective filters applied to each resolution. In (b), the conventional Laplacian of Gaussian filter bank is shown. The Laplacian approximation is applied to the original image data. The signal is recursively filtered and subsampled to obtain each resolution. The frequency responses of the effective filters differ from resolution to resolution in this case.

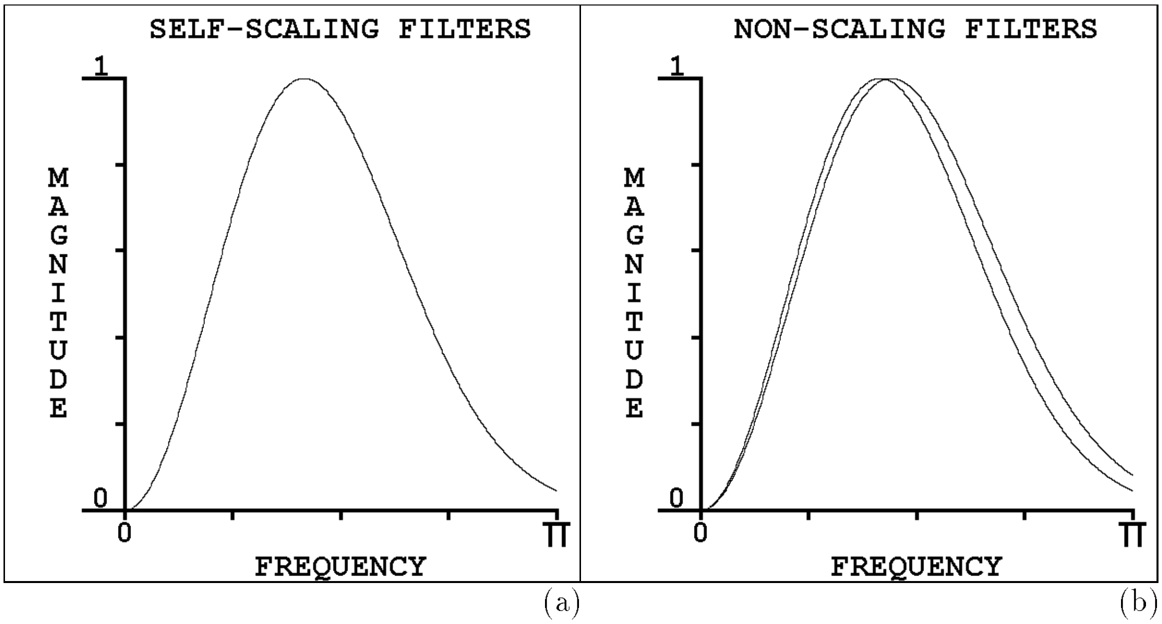


Figure 5-6: Frequency Response of Effective Filters of Laplacian Pyramid. In (a), the frequency response of the effective filters applied to each resolution of the self-scaling Laplacian pyramid. A cross-section of the normalized frequency response is plotted against frequency in radians along the x axis. The frequency response applies to every resolution in the pyramid. In (b), the normalized frequency response is shown for the finest resolution and the adjacent resolution of the non-scaling Laplacian pyramid. In addition to the differences in the spectral shape of the response, the magnitude at the peak differs by a factor of approximately 3.5 from resolution to resolution. Each curve is normalized to the peak in the figure to emphasize the difference in spectral shape.

the alternate approach in which the Laplacian approximation is applied only once, as shown in Figure 5-5b. In that case, the effective filter differs from resolution to resolution. There is a slight discrepancy in the spectral shapes of the filters from resolution to resolution. In addition, the peak magnitude of the effective filter varies by a factor of 3.5 to 4.0 from one resolution to the next, depending on which pair of resolutions is being considered. A comparison of the normalized frequency responses of the effective filters is given in Figure 5-6.

The advantage of a self-scaling pyramid is that subsequent algorithms applied to the data at each resolution are identical when measured in units of the subsampled data. For example, the parameters we use below for smoothing the zero-crossings at different resolutions are the same when measured in units of the subsampled data. Equivalently, the smoothing parameters vary by a factor of two, in absolute terms, from resolution to resolution. This is appropriate because each resolution has been processed with the same effective filter in the subsampled units. In contrast, the alternate approach would require a different smoothing parameter for each resolution, because the effective filter differs from resolution to resolution. This argument extends to other types of processing in addition to contour smoothing.

At each level of the pyramid, the zero-crossings are obtained from the Laplacian of Gaussian signal. The algorithm traces along the border between positive and negative regions. The location of each zero-crossing data point is determined using a bilinear interpolation among a 2×2 neighborhood of the data signal.

The zero-crossings are smoothed using the algorithm described in Chapter 2. The tolerance parameter is chosen empirically to be $\frac{\sigma_i}{8}$, where σ_i is the effective standard deviation of the Gaussian filter for the current resolution. The tolerance parameter is proportional to the spatial width of the filter because the ability to localize the zero-crossings degrades with the amount of smoothing applied to the image[53].

Zero-crossing contours are grouped according to the region they bound. Each region is designated as a positive region or a negative region depending the sign of the values of the signal within the region. For each region there is a single zero-crossing contour that is the exterior boundary and any number of contours that correspond to holes of the region.

For each region, the medial axis skeleton is computed as described Chapter 3. The region decomposition is computed as described in Section 5.3. The result is the set of simple region features for each resolution. The positive simple regions at three resolutions for an image of the word "CODING" is depicted in Figure 5-7.

5.5 Geometric Properties of the Features

Once a region has been decomposed, it is desirable to compute properties of the subregions. In Chapter 6, we shall require the centroid and area of each subregion. We consider these computations here.

As described in Section 2.5.3, it is possible to compute the area of a region directly from the analytical contour representation. In that section, we assume the bounding contour consists entirely of circular arcs. However, the subregions of the decomposition are bounded by line segments as well as circular arcs. Therefore, we must consider the contribution of the line segments to the area and centroid computation.

Recall that we exploit Green's Theorem to convert the area computation from an integral over a region to an integral over the bounding contour. We break the integral into intervals that correspond to circular arcs in the representation. The result is a summation over the circular arcs in the representation. Each term in the summation corresponds to the contribution of the circular arc to the area computation.

From Equations 2.28 and 2.32, we find that when the contour consists entirely of circular arcs

$$A = \sum_{i=1}^{N} A_i = \sum_{i=1}^{N} \frac{1}{2} \int_{s=s_{i-1}}^{s=s_i} \left(x(s) \frac{\partial y}{\partial s}\, ds - y(s) \frac{\partial x}{\partial s}\, ds \right), \tag{5.1}$$

where

$$A_i = \frac{1}{2} \left\{ R x_c \left[\sin\theta\right]_{\theta_{1i}}^{\theta_{2i}} - R y_c \left[\cos\theta\right]_{\theta_{1i}}^{\theta_{2i}} + R^2 \left[\theta_{2i} - \theta_{1i}\right] \right\}. \tag{5.2}$$

We must find a similar expression for A_i when the i^{th} interval corresponds to a line segment, rather than an arc.

The position of a line segment may be expressed as a function of the length along the segment as

$$x(s) = x_1 + s \frac{x_2 - x_1}{l}, \tag{5.3}$$

$$y(s) = y_1 + s \frac{y_2 - y_1}{l}, \tag{5.4}$$

where the points (x_1, y_1) and (x_2, y_2) are the endpoints of the segment, l is the length of the segment, and s is the distance from a point on the segment to (x_1, y_1). Note also that

$$\frac{\partial x}{\partial s} = \frac{x_2 - x_1}{l}, \tag{5.5}$$

$$\frac{\partial y}{\partial s} = \frac{y_2 - y_1}{l}. \tag{5.6}$$

Substitution of these expressions into 5.1 yields

$$A_i = \frac{1}{2} \left\{ \int_0^l \left(x_1 + s \frac{x_2 - x_1}{l} \right) \frac{y_2 - y_1}{l}\, ds - \int_0^l \left(y_1 + s \frac{y_2 - y_1}{l} \right) \frac{x_2 - x_1}{l}\, ds \right\}. \tag{5.7}$$

After evaluating the integral, this expression may be simplified to

$$A_i = \frac{1}{2} \left\{ y_2 x_1 - x_2 y_1 \right\}. \tag{5.8}$$

Similarly, in Section 2.5.4 we find that it is possible to compute the centroid of a region directly from the analytical contour representation. Following the same logic, we use Green's Theorem to reduce a pair of integrals (one for each component of the centroid) over a region to a pair of integrals over a contour. We break each integral into intervals that correspond to the circular arcs in the representation. The result is a summation over the circular arcs in the representation. Each term in the summation is the contribution of a particular arc to the integral.

Again, we must determine the contribution of a line segment to the appropriate contour integral. Recall from Section 2.5.4 that $\overline{x}$, the x component of the centroid, may be written

$$\overline{x} = \frac{1}{A}\sum_{i=1}^{N} B_i = \frac{1}{A}\sum_{i=1}^{N}\int_{s=s_{i-1}}^{s=si} \left[-x(s)y(s)\right]\frac{\partial x}{\partial s}\,ds, \tag{5.9}$$

where A is the area of the region and B_i is the contribution of the i^{th} arc to the integral. For a circular arc,

$$\begin{aligned} B_i \;\; = \;\; & -Rx_c y_c\left[\cos\theta\right]_{\theta_{1i}}^{\theta_{2i}} + R^2 x_c\left[\frac{\theta}{2} - \frac{\sin 2\theta}{4}\right]_{\theta_{1i}}^{\theta_{2i}} - \\ & R^2 y_c\left[\frac{\cos 2\theta}{4}\right]_{\theta_{1i}}^{\theta_{2i}} + R^3\left[\frac{\sin^3\theta}{3}\right]_{\theta_{1i}}^{\theta_{2i}}. \end{aligned} \tag{5.10}$$

To determine the contribution of a line segment to the integral we substitute Equations 5.3, 5.4 and 5.5 into Equation 5.9. This yields

$$B_i = -\int_0^l \left(x_1 + s\frac{x_2 - x_1}{l}\right)\left(y_1 + s\frac{y_2 - y_1}{l}\right)\frac{x_2 - x_1}{l}\,ds. \tag{5.11}$$

After evaluating the integral, this expression may be simplified to

$$B_i = \frac{x_1 - x_2}{6}\left(x_1 y_2 + y_1 x_2 + 2x_2 y_2 + 2x_1 y_1\right). \tag{5.12}$$

Similarly, the y component of the centroid, $\overline{y}$, may be computed

$$\overline{y} = \frac{1}{A}\sum_{i=1}^{N} C_i = \frac{1}{A}\sum_{i=1}^{N}\int_{s=s_{i-1}}^{s_i} x(s)y(s)\frac{\partial y}{\partial s}\,ds, \tag{5.13}$$

where C_i is the contribution to the integral of the i^{th} interval of the integration. In the case where the interval corresponds to a line segment, rather than a circular arc, the contribution may be written

$$C_i = \frac{y_2 - y_1}{6}\left(x_1 y_2 + y_1 x_2 + 2x_2 y_2 + 2x_1 y_1\right). \tag{5.14}$$

The area and centroid of the subregions may be computed analytically from the representation. The computations mirror those for the entire region. However, the portions of the subregions that correspond to line segments rather than circular arcs must be treated separately.

5.6 Discussion

The procedures described in this chapter yield a set of features at multiple resolutions. The features consist of regions with simple shapes. These regions typically correspond to subjective features in the image. More importantly, as we shall see in Chapters 6 and 7, the regions are stable as a sequence of images evolves in time.

In Figure 5-7 an image of the word "CODING" is depicted along with its positive simple regions at three resolutions. At the most coarse resolution (Figure 5-7b), "CODING" corresponds to a single zero-crossing region. At the middle resolution (Figure 5-7c), more of the structure of each individual letter is present, although the letters are still not distinct. At the finest resolution (5-7d), each letter is clearly evident in the zero-crossing pattern.

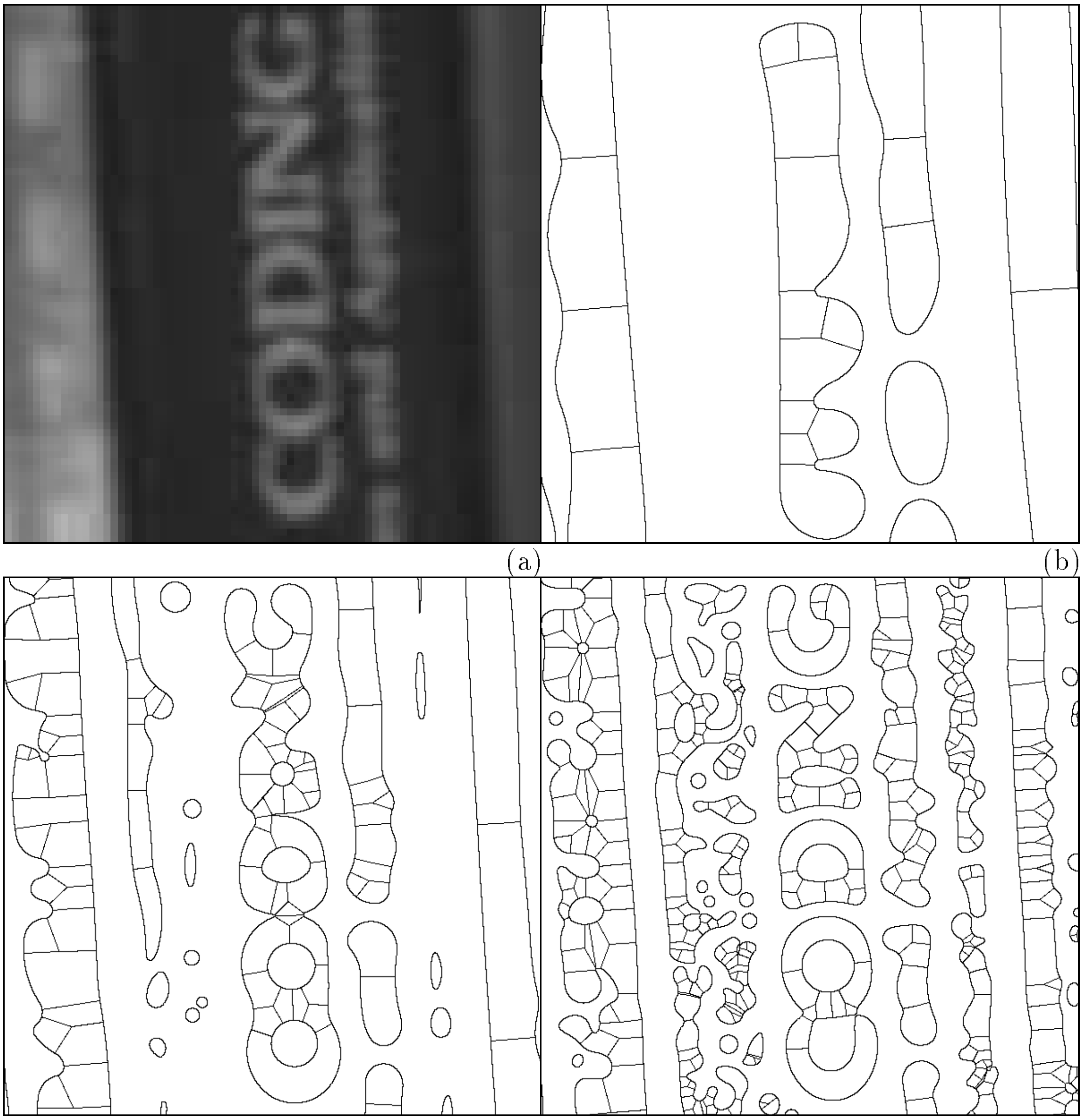
However, even at the finest resolution, some of the letters have merged together. The "C" and the "O" consist of a single region bounded by a single zero-crossing. The same is true for the "I" and the "N". The function of the region decomposition is to break the regions into manageable parts. It is the task of subsequent processes to manipulate the simple regions. For example, an optical character recognition algorithm would be required to determine that particular groups of the simple regions correspond to particular letters.

It is desirable to extract features at multiple resolutions. As we discussed in Part I, there is a tradeoff between the complexity of the representation and the accuracy. For some applications, it is necessary that as much detail as possible be present. For other applications, the reduction in computational complexity achieved by using a more coarse resolution may be worth the reduction in accuracy of the representation.

For example, an optical character recognition system would require the feature representation at the finest resolution depicted in Figure 5-7d. However, the motion estimation algorithm presented in Chapters 6 and 7 uses a more coarse resolution. As we shall see, the motion estimation algorithm achieves reasonable results at the more coarse resolution. Because of the reduction in resolution, the amount of processing required to obtain correspondence of the simple regions from frame to frame is greatly reduced.

One of the primary advantages of simple region features over other feature extraction paradigms is the spatial support of each feature. Each simple region feature possesses more abstract attributes than other types of features. Simple regions have location (centroid), area, orientation, and shape attributes. In addition, the connectivity with their neighbors is specified. These properties may be exploited by subsequent processes in a variety of ways. For example, in Chapter 6, we exploit the area, shape, and connectivity of the simple regions when we obtain correspondence from frame to frame for motion estimation.

In contrast, edges are specified by their location. Some edge detection algorithms also supply an orientation attribute, but they are typically inaccurate. Less information is contained in each edge than in each simple region.



(a) (b) (c) (d)

Figure 5-7: Simple Region Features. The image of the word "CODING" is depicted along with the simple regions features at three resolutions.

The simple region feature extraction process is also quite stable. A small perturbation in a zero-crossing contour typically leads to a correspondingly small perturbation of the associated simple region. There are two exceptions to this statement. At a critical point in the contour deformation, a minimum in the distance between the contour and the skeleton may be introduced. When this occurs a single simple region splits into two at the location of the minimum. Conversely, if a minimum is eliminated, two neighboring simple regions merge.

The other exception occurs when an additional skeleton branch is introduced into the representation. The critical point in the deformation occurs when the osculating circle of a positive maximum of curvature is tangent to the bounding contour at another point on the contour. (See Section 3.3 for a discussion of the presence of a skeleton branch.) When a new branch is introduced, the affected region splits into three smaller regions.

There is an addition attribute of the decomposition that makes it stable against perturbations in the zero-crossing contours. When a region breaks into two regions from one frame to the next, the break is likely to occur at a minimum in the distance from the contour to the skeleton. Effectively, under continuous deformation of the contour, a split in the region occurs when this distance becomes zero. As a result, when a break occurs from one frame to the next, it is likely that there existed a segmentation of the region in the corresponding location of the previous frame. Despite the fact that a region has broken into two, the set of simple regions changes only slightly.

The number of discrete changes possible in the decomposition is small. As a result, it is not difficult to account for the changes explicitly. For example, when computing the correspondence from frame to frame, it is natural to consider the correspondence of single simple region in one frame with a pair of neighboring simple regions, separated at a distance minimum, in the next frame. Similarly, it is natural to consider the correspondence between a single simple region and a triple of simple regions that are associated with the same skeleton node.

5.7 Summary

In this chapter, we introduce a novel approach for feature extraction. The primary advantage to this approach is that the features consist of regions rather than points, line segments, or curves. As a result, the features possess more abstract attributes than in previous approaches. In addition, the feature extraction process is stable against perturbations in the input data. In subsequent chapters, we demonstrate the utility of this approach for the problem of egomotion and depth estimation.

Chapter 6

Tracking Across Multiple Frames

6.1 Introduction

In this chapter, we consider an algorithm for tracking simple region features across multiple frames of an image sequence. The development of the tracking algorithm facilitates the estimation of the camera motion and the depth of each feature in Chapter 7. Ultimately, however, the ability to track features across multiple frames has additional benefits. The stable tracking of features across multiple frames is likely to be the cornerstone of extending existing algorithms in the temporal dimension.

For example, almost all object recognition paradigms identify objects from a single image. A stable tracking mechanism would be crucial for determining the identity of an object from a sequence of images. The tracking algorithm would facilitate the update of a hypothetical pose from frame to frame. In addition, a tracking algorithm would be necessary to maintain the appropriate mapping from the features in the image sequence to the template in the data base. Without a stable tracking mechanism, it is necessary to recognize objects from scratch each time a new image is obtained.

We are presently interested in obtaining the correspondence of features for the purpose of estimating the camera motion and the depth of each feature. Tracking the features from frame to frame facilitates the computation of the displacement, or equivalently, the velocity of each feature in the image. The velocity measurements are often called the *optical flow* or the *two-dimensional motion field*. The optical flow measurements are subsequently used to estimate the egomotion and the depth.

Obtaining the correspondence mapping of features in a sequence of images is an essential step for all motion algorithms that utilize features. The correspondence problem has been studied extensively (for example, Ullman[77], Tsai[74], Faugeras & Maybank[24], Weng, Ahuja, & Huang[79]). Consequently, many authors assume that it is solved in their analysis (for example, Broida & Chellappa[9], Young & Chellappa[82], Burger & Bhanu[11], Shariat & Price[70]). However, the correspondence problem remains unsolved, in general. Before proceeding with the recovery of motion and depth estimates, we must develop a tracking mechanism that is applicable to simple region features.

It is desirable to obtain the correspondence of features over multiple frames because doing so makes the motion and depth estimation more accurate and robust. It is well-known that increasing the distance that the camera moves between frames (i.e. the baseline) in a two-frame motion estimation algorithm improves the accuracy of the motion and depth estimates (for example, Shariat & Price[70], Heel[30], Michael[56], or Okutomi & Kanade[58]). However, increasing the baseline also has the effect of increasing the distortion of the image from one frame to the next frame. Therefore, determining correspondence mapping becomes more difficult.

Obtaining a correspondence across multiple frames is beneficial for both of these considerations. First, extending the track in time increases the total displacement of the camera, thereby increasing the effective baseline. The subsequent computations of the depth estimates are then more robust. Second, by propagating the track from frame to frame the correspondence is obtained over relatively small increments. Between each frame, the distortion of the image is small compared to the distortion that would occur over the effective baseline. Therefore, propagating the correspondence mapping from frame to frame is more reliable than obtaining the correspondence from the two end frames exclusively.

While it is advantageous to obtain the correspondence over multiple frames, it is not desirable to retain all of the displacement measurements. Doing so would require substantial memory that may not be available. Instead, it is preferable to combine the estimates recursively and to maintain only the current state of the recursive estimator in memory, rather than all of the measurements.

In this chapter, we consider a method for tracking simple region features across multiple frames. The first step is to determine the correspondence mapping of features in adjacent frames of an image sequence. Propagating the correspondence mappings over the entire sequence yields the desired tracking capability.

In addition, we develop a recursive estimator that combines the optical flow measurements over time. Therefore, it is not necessary to retain the entire process history of every feature in memory. The recursive estimator provides the capability to improve the flow estimates over time without the necessity of retaining all of the measurements in memory.

The recursive estimator also provides a measure of the accuracy of the optical flow estimate. The measure is based on the stability and persistence of the associated feature. We subsequently exploit this measure when we estimate the camera motion and the depth. In particular, we give greater weight to the flow measurements that are deemed to be more reliable. Furthermore, the accuracy measure for the optical flow also provides an accuracy measure for each depth estimate.

Many of the desirable properties of the multiple frame tracking algorithm are a direct result of the attributes of the simple region feature extraction paradigm. The algorithm for obtaining the correspondence mapping between two frames exploits the shape, size, and connectivity of the features. The measurement of the displacement between corresponding features is robust due to the fact that the features are regions rather than points. The recursive estimator further exploits the shape and connectivity attributes of the features to determine the reliability of a particular feature correspondence.

6.2 Correspondence Between Two Frames

The correspondence algorithm considers simple regions obtained from two adjacent frames in the image sequence at a particular resolution. To differentiate these frames, we call one the previous frame and the other the next frame. The objective is to find a mapping from the features in the previous frame to the corresponding features in the next frame.

The general approach to finding the correspondence of simple regions in adjacent frames is to define a function that measures the desirability of the mapping. This optimization function includes a local measure that depends on the similarity of corresponding features. In addition, the optimization function includes a measure that depends on the smoothness of the optical flow between neighboring features.

There are many possible choices for the local similarity measure. The measure could be a function of the geometric properties of the features such as the area, orientation, and elongation of corresponding features. Alternatively, the similarity measure could be a based on a shape metric or even a symbolic description of the bounding contour of the feature. The key is to devise a measure that is near zero when corresponding features are similar and large when the features are not.

For simplicity, we choose the difference of the area between simple regions as the local measure of the similarity of the features. That is,

$$L_i = |A_i - A_j|, \tag{6.1}$$

where L_i is the similarity measure of a feature and its corresponding feature, A_i is the area of the feature of interest, and A_j is the area of the corresponding feature. Of course, this measure ignores several important characteristics of the shapes of regions. However, even this simple measure provides an enormous benefit for determining the correspondence.

The smoothness of the optical flow is equivalent to the overall deformation of features from one frame to the next. As a sequence evolves, the image typically does not change drastically from frame to frame. As a result, the change in the relative positions of features within the image is typically small. This assertion is equivalent to stating that the optical flow is typically smooth.

We choose a smoothness measure that depends on the correspondence mapping of a feature and the mappings of its neighbors. The smoothness measure for a particular feature is the sum of the squared magnitude of the difference between the flow of the features and that of its neighbors. More specifically, the smoothness measure associated with a particular feature is given by

$$S_i = \sum_k (u_i - u_k)^2 + (v_i - v_k)^2 \,, \tag{6.2}$$

where S_i is the smoothness measure pertaining to a particular feature, $(u_i, v_i)^{\mathrm{T}}$ is the optical flow of the feature, and $\left\{(u_k, v_k)^{\mathrm{T}}\right\}$ are the optical flow vectors of the neighbors.

The smoothness measure prefers that the difference in the optical flow of neighboring features be zero or at least small. Note that the measure does not act to smooth the optical flow after the correspondence has been found. Rather, the measure causes the correspondence algorithm to prefer mappings that result in a smooth flow field.

The individual optimization measure for a particular feature is the sum of the similarity measure and the smoothness measure. The total optimization measure for the image pair is the sum of the individual optimization functions over all features. The total optimization measure prefers smooth correspondence mappings where the individual features correspond to features that are similar in size. This optimization measure is the basis for determining the correspondence mapping between two successive frames in the sequence.

Given the optimization measure, it is necessary to determine the mapping that optimizes the function. Unfortunately, there is a combinatorical explosion related to the possible number of pairings from one frame to the next. To circumvent this problem, we shall devise a strategy in which only a small fraction of the space of possible correspondences is considered. Of course, we desire a strategy that is likely to consider the optimal correspondence mapping.

The strategy for enumerating the possible correspondences may be viewed as a heuristic search of the correspondence space. Under this strategy, the correspondence is initialized locally under a constraint imposed by the shape of the features. As the correspondence algorithm progresses, the spatial extent considered by the algorithm is increased and the shape constraint is relaxed. In the process, the correspondence mapping obtained in the initial stages of the algorithm propagates throughout the regions bounded by zero-crossings.

The heuristic search strategy has three main processing stages. In the first stage, the correspondence mapping is initialized using only the local optimization measure, L_i. A shape constraint that is described below is imposed during the first processing stage. In the second processing stage, the algorithm considers groups of features and uses the total optimization function. The groups of interest are the simple region features that correspond to a single branch and the triples of features that share a common skeleton node. In the second stage, the same shape constraint is imposed on the individual features. In the third processing stage, the algorithm considers groups of features again using the total optimization function. However, the shape constraint is not imposed.

The shape constraint employed in the initial processing stages is based on a classification of each simple region into one of nine shape classes. Each simple region has two ends. At each end, the simple region may have zero, one, or two neighbors. Since there are three possibilities for each end, there are nine classes of shapes as defined in this manner. The nine shape classes are illustrated in Figure 6-1.

It is worth noting that groups of features may also fall into one of the shape classes. For example, consider a small perturbation of the bounding contour of a simple region feature. Suppose that the perturbation causes the simple region to split into three because an additional branch is introduced into the skeleton representation. The three smaller simple regions, when taken as a group, fall into the same shape class as the original simple region. Similarly, a deformation of a simple region that introduces a distance minimum causes the simple region to split into two. The resulting pair of simple regions belongs to the same shape class as the original. These effects are illustrated in Figure 6-2.

In the first and second processing stages, the correspondence algorithm only considers mappings that preserve the shape class of a feature or combination of features. The shape constraint greatly reduces the computational burden of the algorithm. The constraint eliminates a large fraction of the potential correspondence mappings from consideration.

The first processing stage consists of three parts. In the first part, initial mappings from each feature in the previous frame are made to features in the next frame. In the second part, initial mappings from each feature in the next frame are made to features in the previous frame. The first and second processing parts are identical except that the roles of the two frames are interchanged. In the third part, mappings from the first two parts that disagree are rectified.

The mappings of parts one and two are determined in the following manner: A small, arbitrary area around the centroid of a particular feature is considered. The features from the other frame that fall within the area of interest are taken as the initial candidates for correspondence. Candidate mappings that do not preserve the shape class of the feature of interest are not considered. The candidate mapping that minimizes the local optimization function is chosen for the initial correspondence of the feature of interest.

As illustrated in Figure 6-2, a single simple region feature in one frame may correspond to more than one simple region in the other frame. It is necessary to consider this possibility when choosing the correspondence mapping. Therefore, combinations of features must be considered as candidates for correspondence with the feature of interest.

Consider the correspondence of a single feature in one of the frames. Assume that there are N candidate features in the search area of the other frame. Order 2^N combinations of the candidate features exist. However, all of these combinations need not be considered. Due to the shape constraint, only combinations that consist of

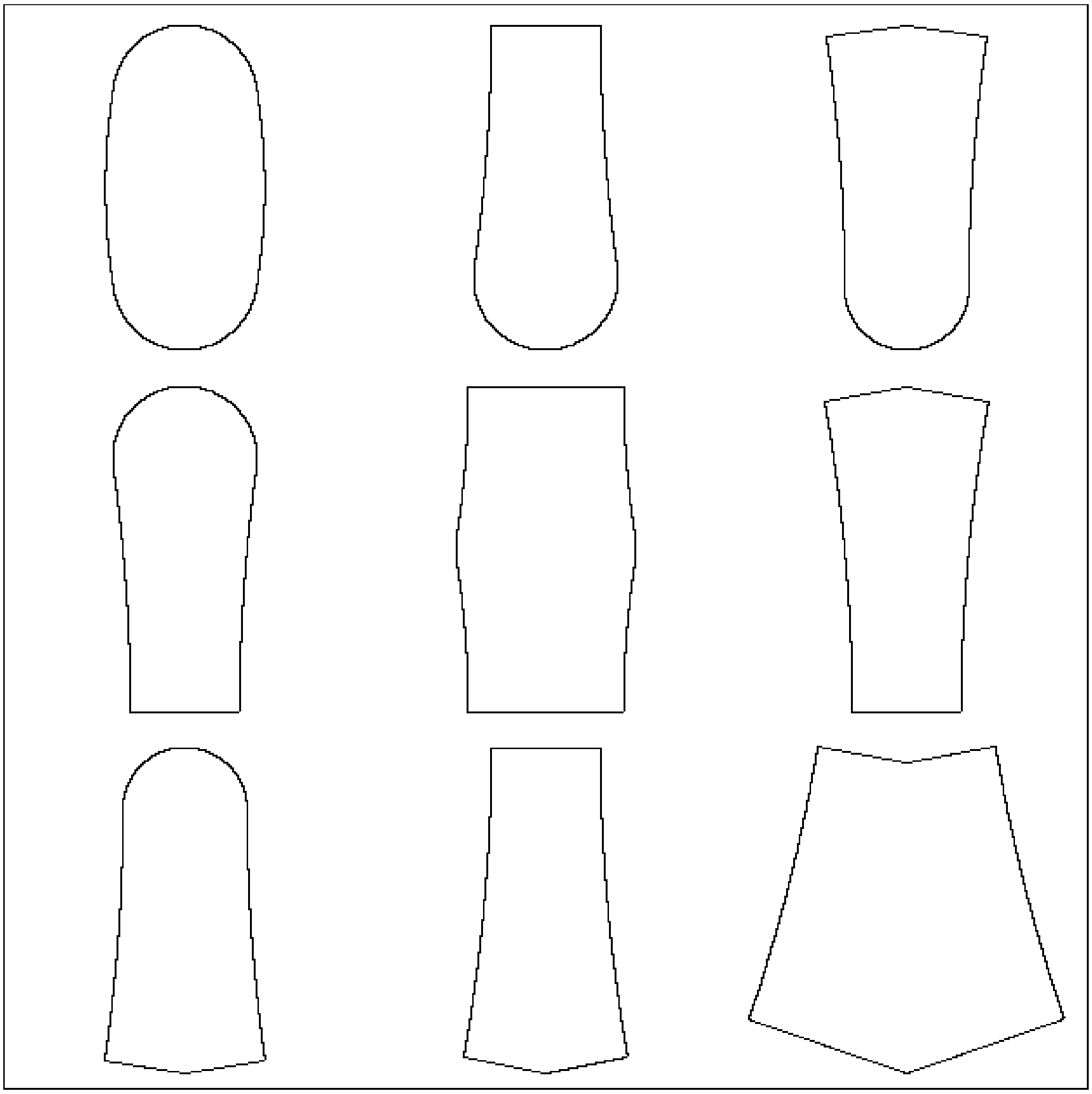
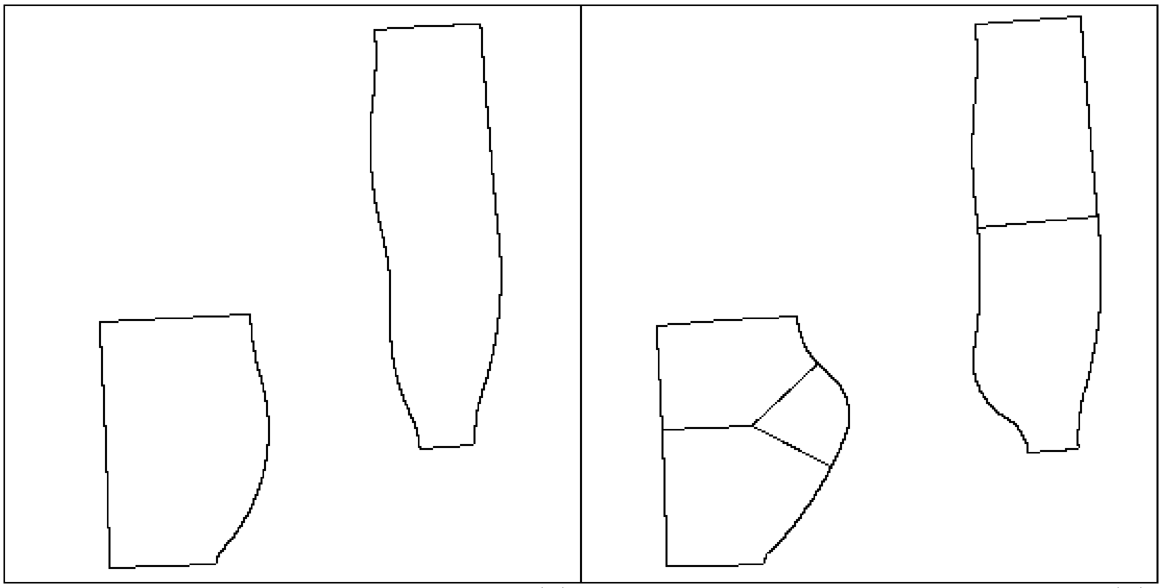


Figure 6-1: Shape Classes. This figure depicts exemplars of the nine shape classes that are used by the shape constraint. In the top row, each simple region feature has zero neighbors on one end (the bottom end). In the second row, each feature has one neighbor on the bottom end. In the third row, each feature has two neighbors on the bottom end. Similarly, the left column contains features with zero neighbors on the top end, the center column has one neighbor on the top end, and the right column has two neighbors on the top end.



(a) (b)

Figure 6-2: Feature Split. At critical occurrences, a simple region feature split into two or three smaller features. In (a), two simple region features are depicted. In (b), a small perturbation in the bounding contour has caused each feature to break into smaller features. The left feature breaks into three because an additional branch has been added to the skeleton representation. The right feature splits into two because a minimum in the distance between the contour and the skeleton branch is introduced. In each case, the resulting combination of features retain the same shape class as the original feature.

contiguous groups of features need to be considered. Furthermore, these contiguous groups must fall into the same shape class as the feature of interest.

We seek to exploit the reduced computational burden provided by the shape constraint. The key is to devise an algorithm that enumerates only those combinations that result in a legal shape. One method for achieving this goal is to build combinations of features end to end. That is, we start with a feature that serves as one end of the combination. We add neighbors recursively to the combination until a termination condition is found. During this process, combinations that result in the desired shape are considered for the correspondence mapping.

The algorithm for building combinations end to end is order N^2. Each candidate feature is considered as the end for potential combinations; there are N such features. For each feature, combinations are constructed by adding neighbors end to end; this is a linear operation. An order N^2 algorithm is substantially less burdensome than an order 2^N algorithm. The shape constraint provides an enormous increase in the efficiency of the algorithm.

There is no guarantee that the mappings obtained from the previous frame to the next frame will be identical to the mappings obtained from the next frame to the previous frame. In the third part of the first processing stage, disagreements from the first two parts are rectified. Mappings that agree are retained. Those that disagree are compared based on the local optimization measure. The preferable mapping is retained, the other discarded. The first stage of the correspondence algorithm is illustrated in Figure 6-3.

In the second stage of the correspondence algorithm, groups of features are considered rather than individual features. Some of the groups consist of simple regions from the same skeleton branch. Other groups consist of triples of simple regions that have a skeleton node in common.

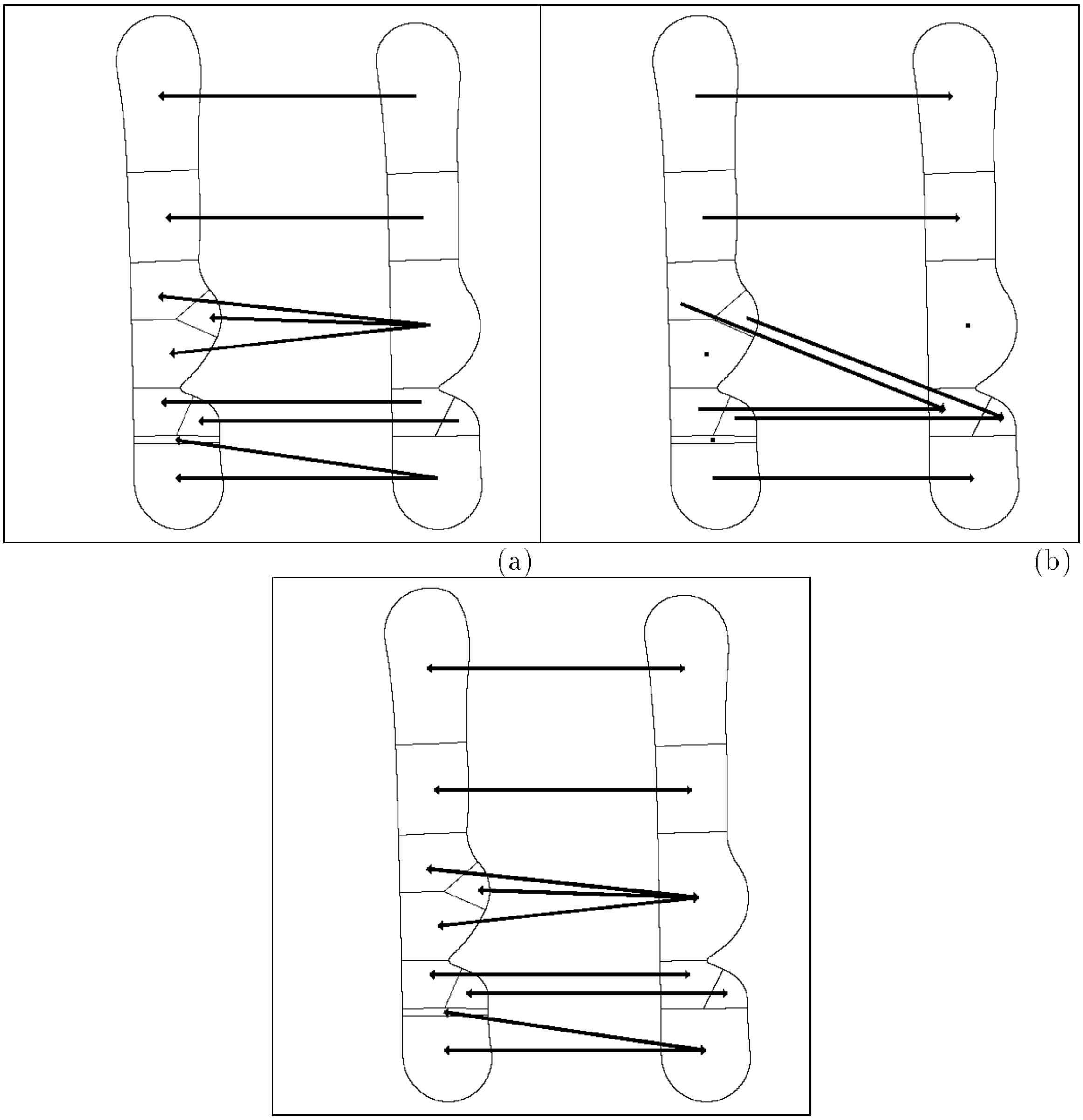
For each group of features considered, the algorithm determines if there are any inconsistencies in the mapping of a feature and its neighbors. That is, if feature A maps to feature B, the neighbor of feature A should map to the appropriate neighbor of feature B. If this is not the case, the mapping of feature A is said to be inconsistent with its neighbor. Similarly, the mapping of feature B is inconsistent with its neighbor.

Whenever an inconsistency between neighbors is found, the algorithm propagates the alternative mappings among the feature group of interest. That is, each mapping suggests mappings for its neighbors. These implied mappings are extended recursively until the entire group of features has a correspondence or until the mapping cannot be propagated further. During the propagation of the mappings, the shape constraint is enforced. No correspondence mappings are permitted between features of different shape classes.

Correspondence mappings are propagated recursively about an initial match by considering the neighbors of corresponding features. Suppose that feature A maps to feature B. This mapping implies that the neighbor of feature A (call it feature A1) maps to the appropriate neighbor of feature B (call it feature B1). If this mapping is acceptable, it is included in the candidate group mapping. In the context of stage two of the correspondence algorithm, A1 and B1 are required to belong to the same shape. If the mapping of A1 to B1 is acceptable, the mapping of their neighbors is considered recursively.

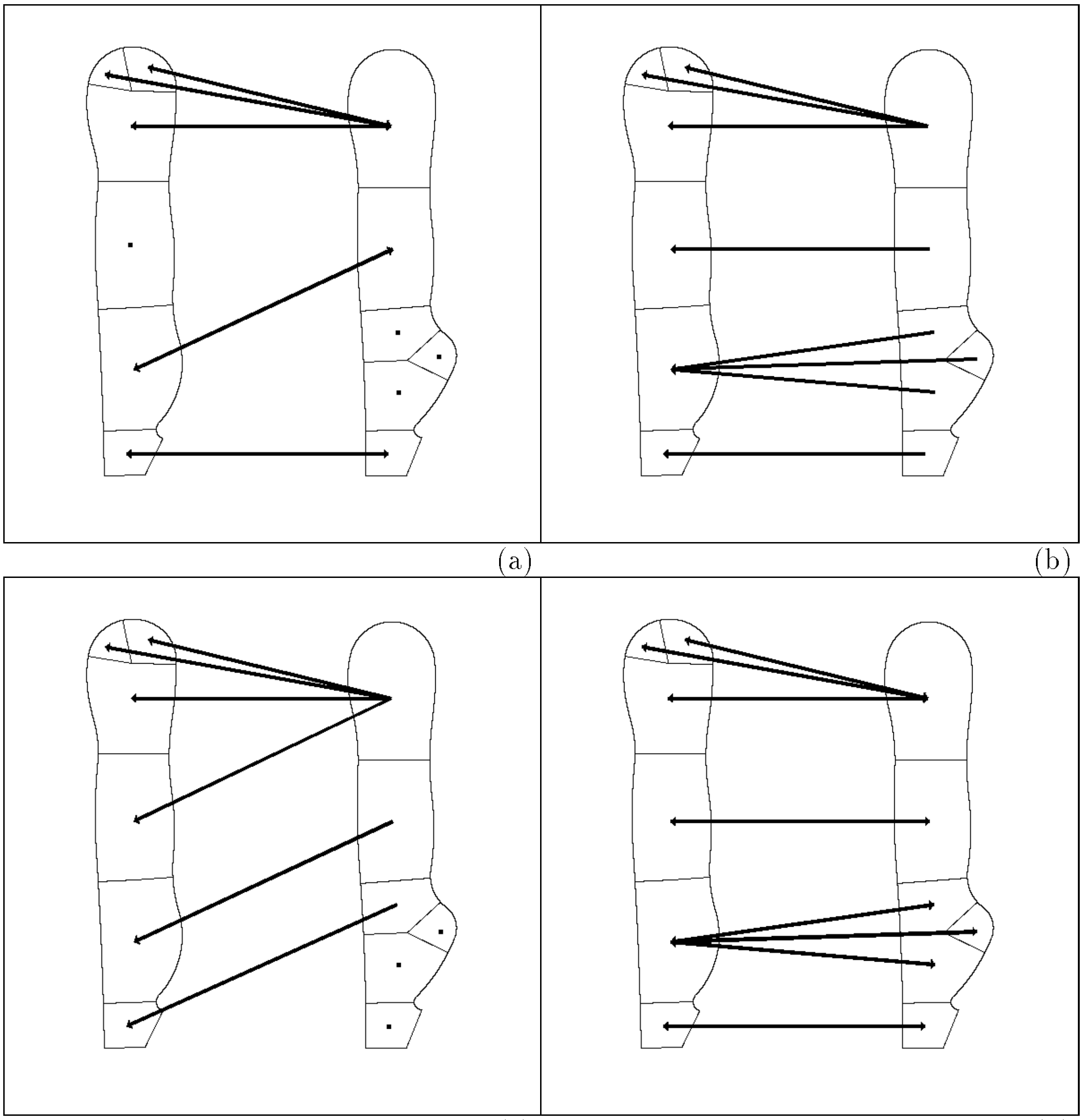
The algorithm extends the alternative group mappings about each inconsistent neighbor. The algorithm chooses from the alternative group mappings by considering the total optimization function for each mapping. The alternative group mapping that minimizes the total optimization function is chosen. The second processing stage of the correspondence algorithm is illustrated in Figure 6-4.

In the third and final stage of the correspondence algorithm, groups of features are again considered. The third processing stage is identical to the second stage, except that the shape constraint is not imposed. The third processing stage is illustrated in Figure 6-5.



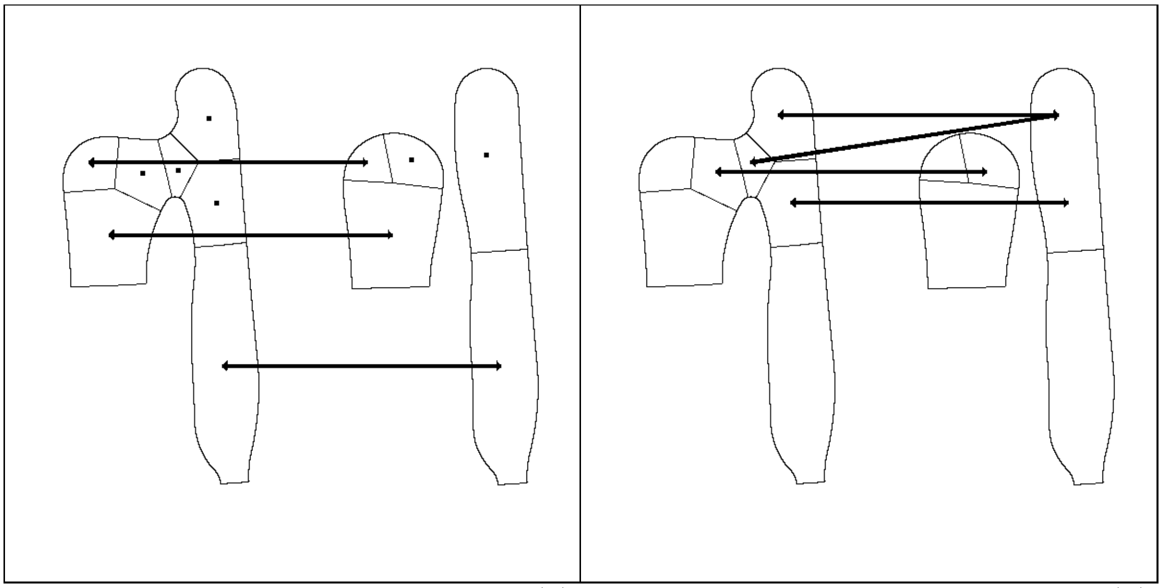
(a) (b) (c)

Figure 6-3: Correspondence Stage I. In stage I of the correspondence algorithm, individual features from one frame are compared to combinations of features in the other frame. For each feature, the combination that belongs to the same shape class and minimizes the local optimization measure is chosen. Stage I consists of three parts. In (a), the first part is illustrated; individual features on the right side are compared to combinations from the left side. The resulting correspondences are illustrated with the vectors. In (b), the second part is illustrated; individual features from the left side are compared to combinations from the right. Again, the resulting correspondences are illustrated. In (c), the third part is illustrated; the correspondence mappings that disagree from parts one and two are rectified. Coincidentally, the mapping of part three is the same as that of part one.



(a) (b) (c) (d)

Figure 6-4: Correspondence Stage II. In stage II of the correspondence algorithm, groups of features are considered. Whenever inconsistent mappings occur between neighbors, the alternative mappings are propagated throughout the group. The mapping that minimizes the total optimization function over the entire group is chosen. In (a), the (incorrect) mapping obtained in stage I is depicted. The top two features on the right side have inconsistent mappings. In (b), the mapping from the top feature is propagated throughout the group. In (c), the mapping from the neighboring feature is propagated throughout the group. In (d), the mapping propagated from the top feature is chosen based on the total optimization function.



(a) (b)

Figure 6-5: Correspondence Stage III. In stage III of the correspondence algorithm, the shape constraint is relaxed. The correspondence mapping is propagated to neighbors that do not have a mapping. In (a), the correspondence mapping obtained in stage II is shown. Several of the features have no match because the appropriate mappings violate the shape constraint. In (b), the additional mappings obtained during stage III are shown. The mappings have propagated from their respective neighboring features.

6.3 Recursive Optical Flow Estimates

Once the correspondence of the features is obtained from one frame to the next, it is possible to measure the displacement of each feature. The displacement is equivalently a measure of the two-dimensional velocity or optical flow of the feature. Because any measurement is imperfect, it is desirable to improve each optical flow estimate by considering multiple measurements over time. However, it is preferable to do so without retaining all of the displacement measurements in memory. A recursive optical flow estimator is described in this section that provides an efficient method to accomplish these objectives.

The optical flow measurement of a feature is the vector difference between the centroid of the feature (or combination of features) and the corresponding feature (or combination) in the previous frame. The difference of centroids is a robust measure of the optical flow. The centroid is stable against small perturbations in the bounding contour. Therefore, even if the shape of the feature changes somewhat from frame to frame, the difference of centroids is very likely to give a reasonable estimate of the flow.

Of course, any measurement contains error. Therefore, the flow measurement is modeled as the true optical flow plus some random vector. Mathematically, we model the flow measurement for each feature as

$$\begin{aligned} u_{mi}[n] &= u_i[n] + r_{ui}[n], && (6.3) \\ v_{mi}[n] &= v_i[n] + r_{vi}[n], && (6.4) \end{aligned}$$

where $(u_{mi}[n], v_{mi}[n])^{\mathrm{T}}$ is the measurement of the optical flow of the i^{th} feature in the n^{th} frame and $(r_{ui}[n], r_{vi}[n])^{\mathrm{T}}$ is a random vector. In addition, we model the optical flow of each feature as being roughly constant over time. The change of the optical flow of a feature from one frame to the next is given by the random vector $(s_{ui}[n], s_{vi}[n])^{\mathrm{T}}$. That is,

$$\begin{aligned} u_i[n+1] &= u_i[n] + s_{ui}[n], && (6.5) \\ v_i[n+1] &= v_i[n] + s_{vi}[n]. && (6.6) \end{aligned}$$

Following well-known results from recursive estimation theory (see, for example, Van Trees[73], Anderson & Moore[2], or Papoulis[60]), we choose a recursive estimator of the form

$$\begin{aligned} \hat{u}_i[n] &= \hat{u}_i[n-1] + K_i[n]\left(u_{mi}[n] - \hat{u}_i[n-1]\right), && (6.7) \\ \hat{v}_i[n] &= \hat{v}_i[n-1] + K_i[n]\left(v_{mi}[n] - \hat{v}_i[n-1]\right), && (6.8) \end{aligned}$$

where $\hat{u}_i[n]$ and $\hat{v}_i[n]$ are the estimates of the optical flow at time n and $K_i[n]$ is a yet unspecified gain. This is the well-known Kalman filter. The filter specified by

Equations 6.7 and 6.8 is simplified relative to the general form based on our assumptions pertaining to the system model and the measurement.

The estimate of the optical flow at time n depends on the estimate from the previous frame, the current measurement, and a gain term. If the gain, $K_i[n]$, is zero, the new estimate is equal to the previous estimate. If the gain is unity, the new estimate is equal to the current measurement. When the gain lies between zero and unity, the new estimate is a weighted sum of the previous estimate and the current measurement.

In order to specify the appropriate gain term, we must make some additional assumptions regarding the random variables in the model. We assume that $r_{ui}[n]$ and $r_{vi}[n]$ are independent random variables, each with variance, $\sigma_{ri}^2[n]$. Similarly, $s_{ui}[n]$ and $s_{vi}[n]$ are independent, each with variance $\sigma_{si}^2[n]$. Assume also that at time $n-1$, estimates have been obtained for the flow, $(\hat{u}_i[n-1], \hat{v}_i[n-1])^{\mathrm{T}}$, and the error variance of each of these estimates is known to be $\sigma_i^2[n-1|n-1]$.

Under these assumptions, the error variance associated with using the previous time optical flow estimate at the current time is given by

$$\hat{\sigma}_i^2[n|n-1] = \hat{\sigma}_i^2[n-1|n-1] + \sigma_{si}^2[n]. \tag{6.9}$$

We call $\hat{\sigma}_i^2[n|n-1]$ the prediction variance because it is the error associated with predicting the optical flow from previous measurements. The notation $[n|n-1]$ refers to the estimate for the optical flow at time n given the measurements up to time $n-1$.

The optimal gain for the recursive estimator is given by

$$K_i[n] = \frac{\hat{\sigma}_i^2[n|n-1]}{\hat{\sigma}_i^2[n|n-1] + \sigma_{ri}^2[n]}. \tag{6.10}$$

The optimal gain may have any value between zero and unity. The gain is near unity when the prediction variance is large compared to the measurement variance. Conversely, the gain is near zero when the prediction variance is small compared to the measurement variance. When the optimal gain is used by the recursive estimator, the variance of the resulting estimate is given by

$$\hat{\sigma}_i^2[n|n] = \hat{\sigma}_i^2[n|n-1] - \frac{\hat{\sigma}_i^2[n|n-1]}{\hat{\sigma}_i^2[n|n-1] + \sigma_{ri}^2[n]}. \tag{6.11}$$

To complete the definition of the recursive estimator, we must specify the initialization of the estimator. The first estimate of the optical flow of a feature is given by the first measurement. The variance of the estimate is simply the measurement variance.

Equations 6.7 through 6.11 specify a recursive method for estimating the optical flow and the variance of the flow from a series of measurements. Under the assumptions outlined in this section, the estimator provides the linear least square-error estimate of the optical flow of each feature from a sequence of measurements. In addition, the recursive estimator provides the variance of each flow estimate.

Unfortunately, the prediction variance and the measurement variance are not known quantities. As a result, it is not possible to obtain truly optimal estimates of the optical flow. Despite this fact, the recursive estimator does provide a useful method for combining a sequence of flow measurements.

In order to make use of the recursive estimator, we must specify the values of the prediction and measurement variances. To make a reasonable choice, we must consider desirable properties of the estimator. It is desirable that the variance measure provided by the recursive estimator reflect the confidence in the optical flow estimate. Therefore, if there is some reason to believe that a particular measurement is better than another, the choice of measurement variances should reflect this hypothesis. In addition, if there is some reason to believe that the prediction of the optical flow from the previous estimate is incorrect or relatively inaccurate, this hypothesis should be reflected in the choice of the prediction variance.

With this in mind, the features are divided into two categories based on their correspondence mapping. For lack of better terminology, we call these categories the *preferred* features and the *common* features. As suggested by the name, we give the variances associated with the preferred features lower values than the variances associated with the common features.

A preferred feature is one that has a mapping from the previous frame that satisfies three criteria. First, the feature must map to a single feature with the same shape. Second, the mapping of the feature must be consistent with the mapping of its neighbors. Third, each neighbor must map to a single feature with the same shape. If a feature does not meet the criteria to be preferred, it is common.

A preferred feature is likely to have a more accurate optical flow measurement than a common feature. The classification test for features is essentially a test of the stability of the feature and its neighbors from one frame to the next. The measurement error associated with a stable feature is likely to be significantly less than the error associated with an unstable feature. In addition, a preferred feature is less likely to have an incorrect correspondence mapping than a common feature. The decreased possibility of an incorrect correspondence mapping for a preferred feature should be reflected in a decreased prediction error.

In the present implementation, the measurement and prediction variances of a preferred feature are given the values 1.0 and 0.1, respectively. If a feature is common, the measurement and prediction variances are given the values 10. and 1.0, respectively. The units of the variance measures are length squared with a unit length being equal to the spacing of the picture cells.

The distinction between the preferred and common features is somewhat arbitrary. The choice of the specific values of the variance parameters for each category is also arbitrary. The specification of the variance parameters requires additional study.

However, the specification of the variance parameters defined above provides us with the necessary means to consider the utility of the recursive estimator. As we shall see in Chapter 7, the recursion provides substantial improvement to the optical flow estimates despite the sub-optimal specification of the variance parameters. There is also a strong correlation between the variance provided by the estimator and the quality of the estimate. These observations demonstrate that recursion is a powerful tool in this context.

6.4 Discussion

In this chapter, we have considered a method for determining the correspondence mapping between simple region features in adjacent frames. Based on the correspondence, we obtain a displacement measurement that is the vector difference of the centroids of corresponding features. A recursive estimator combines the displacement measurements over a sequence of images. At each frame, an estimate of the optical flow and a measure of the accuracy of the flow are provided for each feature.

The algorithms presented in this chapter are formulated to demonstrate the utility of the feature extraction process described in Chapter 5. They are not intended to be the final, optimal methods for determining correspondence and for propagating the displacement measurements in time. Rather, they are intended to demonstrate some of the advantages inherent in the simple region feature extraction paradigm.

For example, the local optimization measure from Section 6.2 is far from the best method of comparing two simple shapes. (Recall that the measure of similarity is simply the difference of the area.) However, using this measure demonstrates that the geometric properties of the features yield a substantial benefit for the processing. Development of more sophisticated similarity measures that include the orientation and elongation of the feature, for example, will improve the stability and efficiency of the correspondence algorithm.

In contrast, consider the correspondence of Canny edges from one frame to the next. A local measure of similarity is almost meaningless in this context. Taken as individual features, Canny edges possess only a crude orientation attribute, in addition to their position. Therefore, any measure of the desirability of correspondence must be based on some global (smoothness) measure.

Furthermore, any correspondence technique that may be applied to edges may also be applied to simple region features. Indeed, the simple regions are constructed from zero-crossings of the Laplacian of Gaussian filter. The zero-crossings correspond to edge locations. In this sense, the simple region features subsume edges.

The shape constraint imposed in the early stages of the correspondence algorithm provides another example of an advantage of simple region features. The shape constraint reduces the computational complexity of the correspondence initialization. The reduction in complexity is a direct result of the geometrically abstract nature of the features. That is, the features have shape and size attributes to exploit.

The displacement measurements of the features are robust because the centroids of the regions are used, rather than the edge locations. Consider a simple region that deforms slightly from one frame to the next. The displacement measurement that is obtained may be considered to be the sum of two components: the true displacement and the change in the centroid due to the shape distortion. The position of the centroid typically changes much less than the perturbation in the position of the contour. Therefore, displacement measurements based on the positions of the contours or edges are less reliable than those based on the centroid.

The aperture problem is also greatly reduced by using simple region features rather than edges. The aperture problem is the ambiguity in the optical flow measurement in the direction perpendicular to the brightness gradient (see, for example, Horn[41], p. 283). The aperture problem derives its name from the observation that it is impossible to determine the component of motion parallel to an edge when the edge is observed over a small area (a small aperture). The aperture problem is reduced for simple region features because each feature spans a non-zero area of the image. The "aperture" for each feature is essentially the size of the feature. In addition, the shape of each feature provides greater context within which to disambiguate the optical flow estimate.

The geometric information available for simple regions provides an ability to estimate the accuracy of the optical flow measurement. In Section 6.3, we use an *ad hoc* approach to separate the features into two classes: a preferred class and a common class. The classification is based on the stability of the correspondence mapping. To be classified as preferred, the shape of the feature and its neighbors must be preserved by the correspondence mapping.

As we shall see in Chapter 7, even this *ad hoc* classification has substantial benefits. The classification is the basis for specifying the reliability of the optical flow measurements. Consequently, in Chapter 7, we give greater weight to the more reliable measurements in our estimation of egomotion. In addition, the reliability measure associated with each flow measurement also provides a reliability measure of each depth estimate.

Of course, given the benefits from the *ad hoc* reliability classification, it is desirable to consider a more principled approach to specify the stability of features and to determine the reliability of their optical flow estimates. The variance parameters provided to the recursive estimator should be based on a more sophisticated similarity measure of corresponding features. The development of such an approach warrants additional study. A better understanding of this issue will lead to improved performance of the recursive estimator and the subsequent processes that depend on it.

The variance provided by the recursive estimator may also be viewed as a measure of the persistence and stability of the associated feature. Consider the process history of a particular feature. Suppose that for some number of frames, the correspondence mappings associated with the feature results in a preferred classification. Following the corresponding features from frame to frame, one finds that the variance measure specified by the recursive estimator decreases monotonically. The variance measure decreases from the initial measurement variance and asymptotically approaches a known steady-state value. Therefore, the variance is a measure of the length of time that the feature has persisted in the sequence. If the feature does not have a preferred mapping throughout its entire process history, the variance is a measure of the stability as well as the persistence of the feature.

6.5 Summary

In this chapter, we consider an algorithm for determining the correspondence of simple region features from two frames. The displacement of corresponding features is measured by the vector difference of the centroids. We consider a recursive approach for improving the optical flow estimates from the sequences of displacement measurements. An *ad hoc* classification scheme facilitates the computation of a reliability measure of the flow estimates.

At each step in the computation of the flow estimates, we exploit the geometric properties of the simple region features. We exploit the area, shape, and connectivity of the features when we determine the correspondence between adjacent frames. We further exploit the shape and the connectivity of the features when we estimate the reliability of the displacement measurement.

The displacement measurements, by themselves, do not provide information that is directly useful for interacting with the environment. In the next chapter, we shall determine the motion of the camera relative to the environment and estimate the depth of objects in the environment from the displacement measurements. Such information is essential for a variety of navigational tasks. The ultimate test of the success of the tracking algorithm and displacement measurements is the ability to determine the motion and depth estimates accurately.

Chapter 7

Passive Navigation

7.1 Introduction

Navigation requires the ability to determine the motion of a vehicle and the position of obstacles in the environment relative to the vehicle. It is desirable to perform this task without the use of an active sensor. More specifically, it is preferable to avoid emitting energy from a sonar, radar, laser, or some other sensor into the environment. Therefore, we seek to develop the ability to navigate using a passive sensor: an optical camera.

The general problem of passive navigation has been the focus of a flurry activity in recent years. A number of paradigms have been proposed for attacking the problem. Many reviews on the subject exist, including those contained in Aggarwal & Nandhakumar[1], Hildreth & Ullman[34], Shariat & Price[70], and Heel[30]. Its solution may be seen as an end in itself. However, we consider passive navigation as a means of demonstrating the viability of the simple region feature extraction paradigm.

In particular, we consider the special case of pure translation. The position of the camera changes from frame to frame. However, the direction of the aim of the camera remains constant. In addition, we impose the static environment constraint. It is assumed that objects in the environment do not move with respect to one another.

In this chapter, we consider a method for determining the egomotion of the camera that is specifically designed to use the optical flow estimates obtained in Chapter 6. We employ a weighted least square-error approach to estimate the direction of translation of the camera and the relative depth of the features. The variance provided by the recursive optical flow estimator is the basis of the weighting scheme. Features with lower variance measures are given greater weight in the motion estimation algorithm. The variance of each optical flow estimate also provides a measure of the reliability of the subsequent depth estimate for the feature.

7.2 Technical Prerequisites

We shall use a viewer-centered coordinate system, following Horn[41]. The coordinate system is fixed with respect to the camera. The origin is coincident with the center of projection of the camera. The X-Y plane is parallel to the image plane. And, the Z axis is parallel to the optical axis of the camera.

Consider the motion of the camera with respect to a static environment. The motion of the camera is specified by the velocity of the center of projection of the camera and the rotation of the camera about the center of projection. For simplicity, we shall assume that the rotational component of the motion is zero.

The translational velocity of the camera is specified by the vector $\mathbf{T} = (U, V, W)^{\mathrm{T}}$, where U, V, and W are the X, Y, and Z coordinates of the velocity, respectively. The velocity of a point P with respect to the viewer-centered coordinate system is given by $\mathbf{V} = -\mathbf{T}$. More explicitly

$$\dot{X} = -U, \tag{7.1}$$
$$\dot{Y} = -V, \tag{7.2}$$
$$\dot{Z} = -W. \tag{7.3}$$

Assuming perspective projection, the projection of a point in space, $(X, Y, Z)^{\mathrm{T}}$, onto the image plane is given by

$$x = f\frac{X}{Z}, \tag{7.4}$$
$$y = f\frac{Y}{Z}, \tag{7.5}$$

where f is the focal length and (x, y) is the image point. Given the motion of the camera and a point in space, it is desirable to determine the motion of the associated image point. Differentiating Equations 7.4 and 7.5, we find

$$u = \dot{x} = \frac{\dot{X}}{Z} - \frac{X\dot{Z}}{Z^2}, \tag{7.6}$$
$$v = \dot{y} = \frac{\dot{Y}}{Z} - \frac{Y\dot{Z}}{Z^2}, \tag{7.7}$$

where u and v are the x and y components of velocity of the image point. Making substitutions from Equations 7.1, 7.2, 7.3, 7.4, and 7.5, we find

$$u = \frac{-U + xW}{Z}, \tag{7.8}$$
$$v = \frac{-V + yW}{Z}. \tag{7.9}$$

Equations 7.8 and 7.9 specify the *two-dimensional motion field* of the image. The 2D motion field is the projection of the relative motion of the environment with respect to the camera onto the image plane.

The *optical flow field* specifies the instantaneous velocity of the brightness patterns in the image. Strictly speaking, the 2D motion field differs from the optical flow field. However, under reasonable assumptions, the optical flow field closely approximates the two-dimensional motion field. Therefore, we shall assume that they are identical.

7.3 Weighted Least-Square Error Egomotion

In this section, we consider a weighted least-square error approach to recover egomotion in the pure translation case. We follow the formulation of Bruss and Horn[10]. However, we customize the approach to be consistent with the estimates provided by the recursive optical flow estimator. In addition, we introduce an arbitrary weighting factor to exploit the reliability measure provided for each feature.

As we found in the previous section, the expected flow field is specified by

$$u = \frac{-U + xW}{Z}, \tag{7.10}$$

$$v = \frac{-V + yW}{Z}, \tag{7.11}$$

where u and v are the x and y components of the motion field at the image point (x, y) in the image and Z is the depth at the point of interest. Therefore, we seek the motion and set of depths that minimize the square-error between the measured optical flow and the expected optical flow from the model. We minimize

$$E = \sum_i w_i A_i \left\{ \left(u_i - \frac{-U + x_i W}{Z_i} \right)^2 + \left(v_i - \frac{-V + y_i W}{Z_i} \right)^2 \right\}, \tag{7.12}$$

where w_i is an arbitrary weight, A_i is the area of the feature, $(U, V, W)^T$ is the unit vector in the direction of the translation of the camera, Z_i, is the depth of the i^{th} simple region feature, (x_i, y_i), is the centroid of the i^{th} feature, and (u_i, v_i) is the estimated flow of the i^{th} feature.

Each term of the summation is weighted by the area of the associated feature. By weighting the summation in this manner, the estimation is robust when a feature splits into smaller features, as described in Section 6.2. For example, suppose that a simple region in one frame breaks into two simple regions in the next frame. The resulting pair of simple regions should be given roughly the same weight as the original feature in the summation. Multiplying each term by the area of the simple region feature accomplishes this because the area of the simple region pair is approximately equal to the area of the original simple region feature.

Note that if the translation vector and all the depth values are scaled by a constant factor, the value of E is unchanged. Physically, this implies that scaling the length dimension of all objects in the environment while simultaneous scaling the speed of the viewer by the same amount leads to an identical flow field. As a result, it is only possible to determine the direction of translation and the relative depth of objects in the environment. Therefore, when computing the translation from optical flow measurements, we specify the translation as a unit vector and the depth measurements as relative to some unknown scale factor. We call this effect the *scale ambiguity* (see, for example, Bruss and Horn[10] or Horn[41]).

A particular manifestation of the scale ambiguity is called the *antipodal ambiguity.* If a solution, (U, V, W) with $\{Z_i\}$ is scaled by -1, the resulting error function is unchanged. Physically, the alternative solution corresponds to motion in the opposite direction and a surface that is reflected through the center of projection. Once the optimal solution is found, it is necessary to distinguish between the two antipodal solutions.

Now, we consider the solution of the weighted square-error minimization problem. First, we seek the values of the depth of each feature that minimize each term of the error summation. Next, we substitute these depth values back into the summation. The substitution yields a minimization problem in only three variables, U, V, and W. Unfortunately, there is no known analytical solution to this minimization problem. Therefore, we must find the minimum numerically.

It is convenient to define

$$\alpha_i = -U + x_i W, \tag{7.13}$$

$$\beta_i = -V + y_i W. \tag{7.14}$$

Given a particular translation vector, T, the expected flow at the image point $(x_i, y_i)^T$ is in the direction of $(\alpha_i, \beta_i)^T$. If the depth is known at that point, the expected flow is given by

$$u_i = \frac{\alpha_i}{Z_i}, \tag{7.15}$$

$$v_i = \frac{\beta_i}{Z_i}. \tag{7.16}$$

We seek the value of Z_i that minimizes the i^{th} term of the summation in Equation 7.12. We differentiate the summation with respect to Z_i and set the result to zero. This operation yields

$$\left(u_i - \frac{\alpha_i}{Z_i}\right)\frac{\alpha_i}{Z_i^2} + \left(v_i - \frac{\beta_i}{Z_i}\right)\frac{\beta_i}{Z_i^2} = 0, \tag{7.17}$$

for each feature. Solving this equation, we find that

$$Z_i = \frac{\alpha_i^2 + \beta_i^2}{u_i\alpha_i + v_i\beta_i}. \tag{7.18}$$

It is convenient to note that by using Equation 7.18, we may write

$$u_i - \frac{\alpha_i}{Z_i} = +\beta_i \frac{u_i\beta_i - v_i\alpha_i}{\alpha_i^2 + \beta_i^2}, \tag{7.19}$$

$$v_i - \frac{\beta_i}{Z_i} = -\alpha_i \frac{u_i\beta_i - v_i\alpha_i}{\alpha_i^2 + \beta_i^2}. \tag{7.20}$$

Substitution of Equations 7.19 and 7.20 into Equation 7.12 yields

$$\sum_i w_i A_i \frac{(u_i\beta_i - v_i\alpha_i)^2}{\alpha_i^2 + \beta_i^2}. \tag{7.21}$$

Given a set of measurements $\{u_i, v_i, x_i, y_i, A_i\}$ and weights $\{w_i\}$, Equation 7.21 is a function of the translation vector, $(U, V, W)^T$. The minimization problem involving many unknowns (the translation vector and the set of depth values), is reduced to a minimization involving three unknowns. Furthermore, because the translation vector is constrained to have unit length, there are only two degrees of freedom in the minimization.

The weighted least-square error approach to the determination of egomotion facilitates considerable flexibility in design. A variety of types of information may be incorporated into the motion estimation algorithm by an appropriate choice of the weights. Here, we explore one of the possibilities.

The variance measure that is provided by the recursive tracking algorithm provides a natural basis for weighting each error term. We choose the inverse of the variance of each flow estimate as the weighting term. That is,

$$w_i = \frac{1}{\sigma_i^2}. \tag{7.22}$$

This choice of weights is suggested by a number of results from stochastic theory. For example, the maximum likelihood estimator of a set of measurements corrupted by Gaussian noise is the weighted average of the measurements with each weight proportional to the inverse variance of the respective measurement. Furthermore, the inverse variance may be interpreted as a measure of the information content of the estimate.

The minimum of the weighted square-error function is found numerically. A numerical gradient descent algorithm with an adaptive step size is used. At each iteration, the weighted square-error function is computed for eight points surrounding the current estimate. The points are the radial projection of a square lattice onto the unit sphere. The center of the lattice is the current estimate. The spacing between points in the lattice varies during the course of the computation.

The point that has the lowest weighted square-error function value is chosen as the new current estimate. If no point has a lower value than the current estimate, the current estimate is retained. If no improvement is found, the spacing of points in the lattice is cut in half and the process is repeated. If an improvement is found, the spacing is doubled. In practice, a maximum spacing is established such that the estimate of the translation vector never changes by more than about five degrees during a single iteration. The iteration terminates when the spacing of the lattice points drops below an arbitrary threshold. The threshold corresponds to an angular difference between the points in the lattice of a small fraction of a degree.

For the first frame pair, the initial translation vector is set arbitrarily. In subsequent frames, the translation vector estimate from the previous frame is used as the initial value. For the results presented in the next section, an initial vector is chosen that is $\frac{\pi}{2}$ radians from the correct answer. Because of the antipodal ambiguity, this is the worst possible initial error. Empirical tests indicate that the adaptive gradient descent is not sensitive to the initial value of the translation vector.

Once the gradient descent algorithm has found a minimum, it is necessary to resolve the antipodal ambiguity. As described above, $(U, V, W)^T$ with $\{Z_i\}$ is equivalent to the solution $(-U, -V, -W)^T$ with $\{-Z_i\}$. However, objects in the environment must be in front of the camera. Points in front of the camera correspond to positive depth. Therefore, the solution that yields the larger number of features with positive depth is chosen between the two equivalent alternatives. The choice between the antipodal solutions is straightforward. Typically, there are only a few outliers with negative depth occurring in the correct solution.

7.4 Experimental Results

In this section, we consider experimental results of the weighted least square-error algorithm. For each sequence, we compare the known translation vector to the translation vector estimates as the sequence evolves. We also consider the depth estimates at the final frame of each sequence.

In Figure 7-1, the first frame of a sequence obtained at the CMU Calibrated Imaging Laboratory is depicted. The sequence consists of ten frames. The motion of the camera between each frame is $T = (2, 0, 6)^{\mathrm{T}}$mm. When normalized to a unit vector, the components of the translation vector become $(0.316, 0, 0.949)^{\mathrm{T}}$.

In Figure 7-2a, the components of the estimated translation vector, normalized to a unit vector, are plotted versus the number of frames elapsed. The correct components are superimposed on the plot. In Figure 7-2b, the angular error between the correct translation vector and the estimated translation vector is plotted versus the number of frames elapsed. Notice that the error in the direction of the translation vector decreases rapidly from its initial value and then settles within a steady-state error bound. The error bound in this case is roughly one and a half to two degrees.

In Figure 7-3, the depth estimates are depicted for a portion of the last frame of the CMU sequence. The region of interest is highlighted in Figure 7-3a. This portion of the image is chosen because the depth is nearly constant within this area. In the other quadrants of the figure, depth estimates associated with the region of interest in the image are plotted against the lateral position in the image. The plot is roughly equivalent to an overhead view of the model town.

In each of these figures the absolute depth of each feature is plotted, in meters. Of course, due to the scale ambiguity, only the relative depth is obtained by the estimation algorithm. However, because the length of the baseline is known *a priori*, the relative depth is converted into absolute depth for display purposes.

Each of the three quadrants representing depth estimates in Figure 7-3 corresponds to a different condition on the variance estimates of the features. In Figure 7-3b, the depth estimates are depicted for the features that are within the region of interest and have smallest possible variance measure. In Figure 7-3c, the depth estimates are depicted for the features within the area of interest and in the top half as ranked by their variance measure are depicted. Finally, in Figure 7-3d, the depth estimates of all the tokens within the region of interest are depicted.

As stated, the features associated with the depth estimates in Figure 7-3b have the smallest possible variance. At each stage of the process history of the feature, the feature has a preferred correspondence. That is, for each image pair, the shape of the corresponding feature is preserved exactly. In addition, for each image pair, the neighbors of the corresponding feature have a consistent correspondence. In the CMU sequence, roughly 19% of the features in the last frame possess the smallest variance.

The features associated with the depth estimates in Figure 7-3c constitute the top half of the features within the region of interest, as specified by the variance measure. The fraction of one half is chosen arbitrarily. These features have persisted for several frames, but not necessarily the entire sequence.

Finally, the depth estimates of all of the features within the region of interest are depicted in Figure 7-3d. In the CMU sequence, roughly 90% of the features in the last frame have a valid correspondence. The others have no correspondence or a flow estimate that yields a negative depth.

Figures 7-4, 7-5, 7-6, and 7-7 follow the same template as described above. Part (a) of each figure highlights a region of interest in the image. Part (b) depicts the depth estimates for features within the region of interest that have the smallest possible variance measure. Part (c) depicts feature within the region that are ranked in the top half by variance. Part (d) depicts the depth estimates of all the features within the region of interest.

In each of these features, the depth estimates corresponding to the most stable, persistent features (subfigure (b)) are very accurate compared to the other depth estimates. The overall accuracy of the depth estimates is slightly degraded as additional estimates associated with less stable, less persistent features are added (subfigures (c) and (d)).

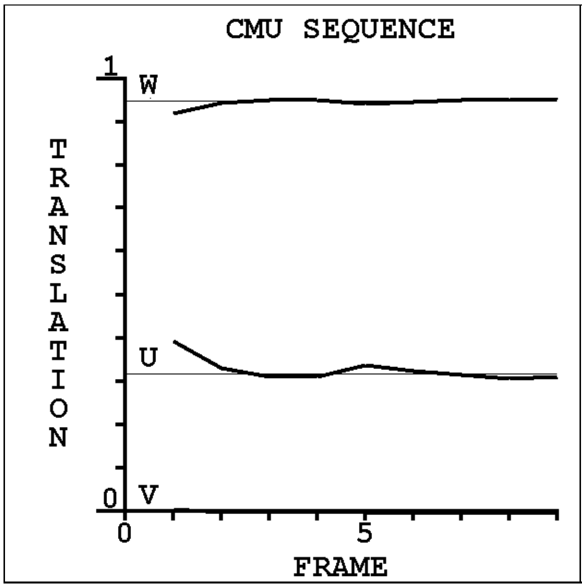
Occasionally, outliers appear in the least reliable depth maps (subfigure (d)). Each depth estimate outlier is the result of an error in the correspondence mapping. When outliers do occur, it is likely that they have a large variance measure because an erroneous correspondence mapping is unlikely to be stable. As a result, a feature that has an incorrect correspondence mapping is unlikely to be classified as preferred.

Figure 7-8 shows the first frame of a different image sequence. The sequence depicts a cup placed in front of a poster and a row of books. The motion of the camera between each frame is $T = (-2, 0, 4)^{\mathrm{T}}$mm. When normalized to a unit vector, the components of the translation vector become $(-.447, 0., .894)^{\mathrm{T}}$. The depth of the cup is known to be 584mm. The depth of the books is approximately 635mm at the far left side of the image. The row of books angle backward toward the center of the image. The depth of the background poster is 914mm (see Heel[30]).

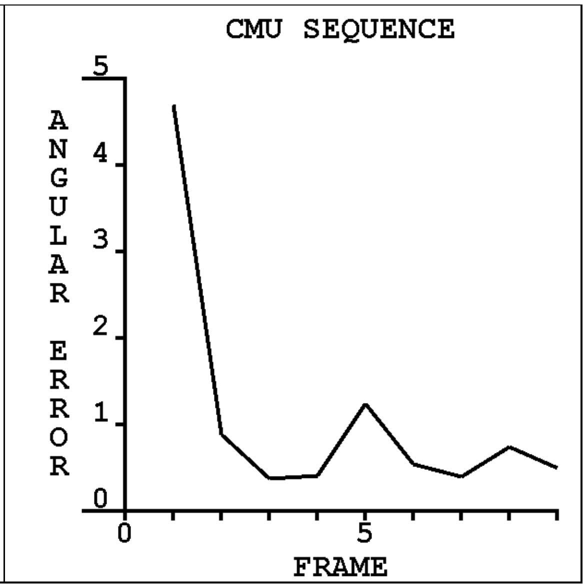
The results of the cup sequence are presented in a similar fashion as the results of the CMU sequence. In Figure 7-9a, the normalized components of the translation vector are plotted versus the number of frames elapsed. In Figure 7-9b, the angular error between the true translation vector and the estimated translation vector is shown. Similar to the CMU sequence, the error in the direction of the translation vector decreases from its initial value and remains within a steady-state error bound. Again, the error bound is within two degrees.

Figures 7-10 and 7-11 illustrate the depth estimates at the last frame of the sequence. In Figure 7-10, the depth estimates of the face of the cup are shown. In Figure 7-11, the depth estimates of the cup and the background are shown. As with the figures pertaining to the CMU sequence, the depths are segregated depending on their associated variance measure.

Figure 7-1: First Frame of the CMU Sequence.



(a)



(b)

Figure 7-2: Translation Vector Estimates. In (a), the normalized components of the estimated translation vector are plotted versus the frame number of the CMU sequence. The correct values, $(0.316, 0, 0.949)^{\mathrm{T}}$, are superimposed on the plot. In (b), the angular error between the estimated translation vector and the correct translation vector is plotted versus the frame number of the sequence.

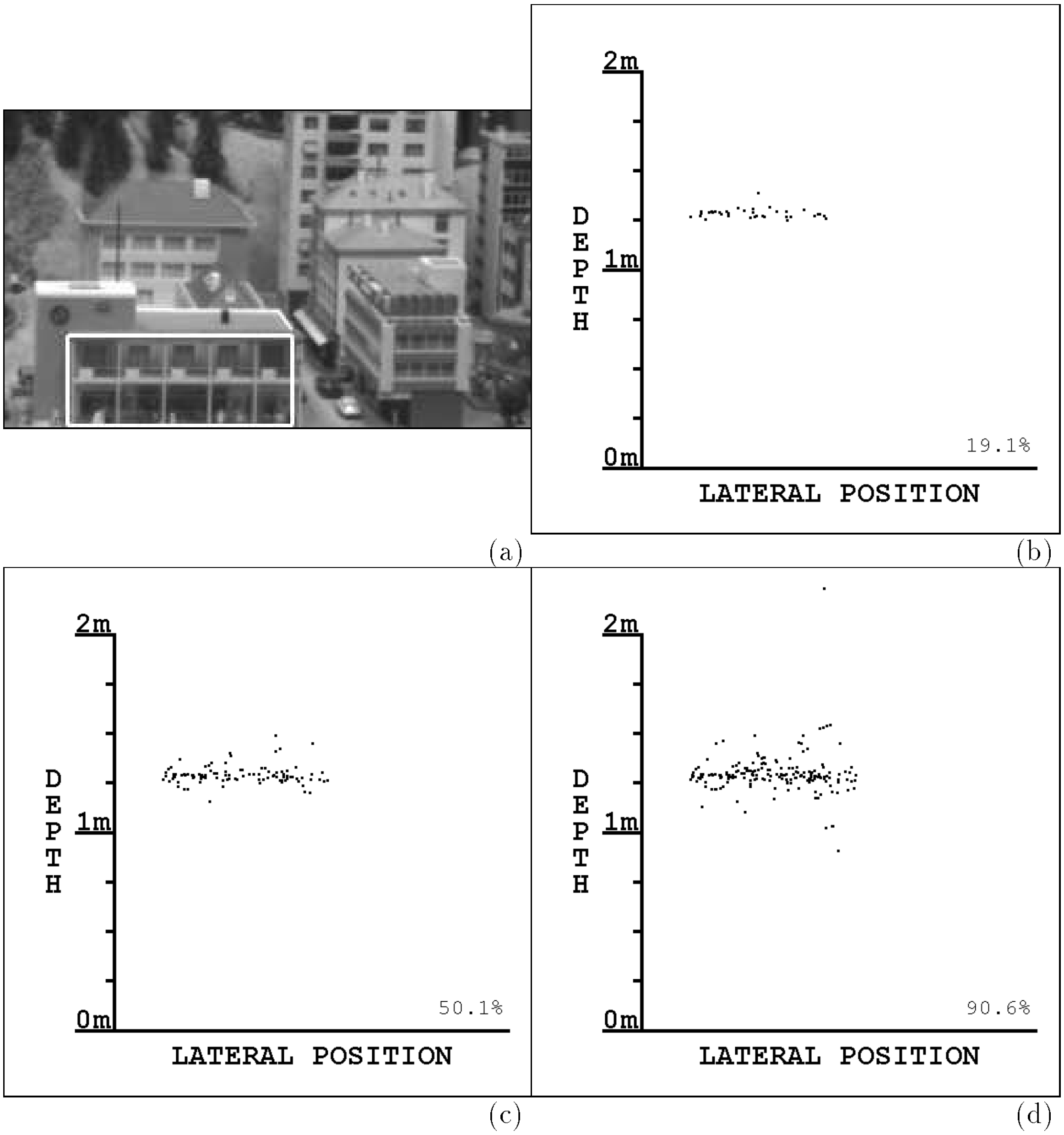


Figure 7-3: Depth Estimates for CMU Sequence. In (a), a region of interest from the last frame of the CMU sequence is outlined. In (b), depth estimates of the most stable and persistent features within the region of interest are plotted versus horizontal position in the image. In (c), depth estimates from the region of interest that rank in the top half based on the variance measure are plotted. In (d), depth estimates for all the features within the region of interest are plotted.

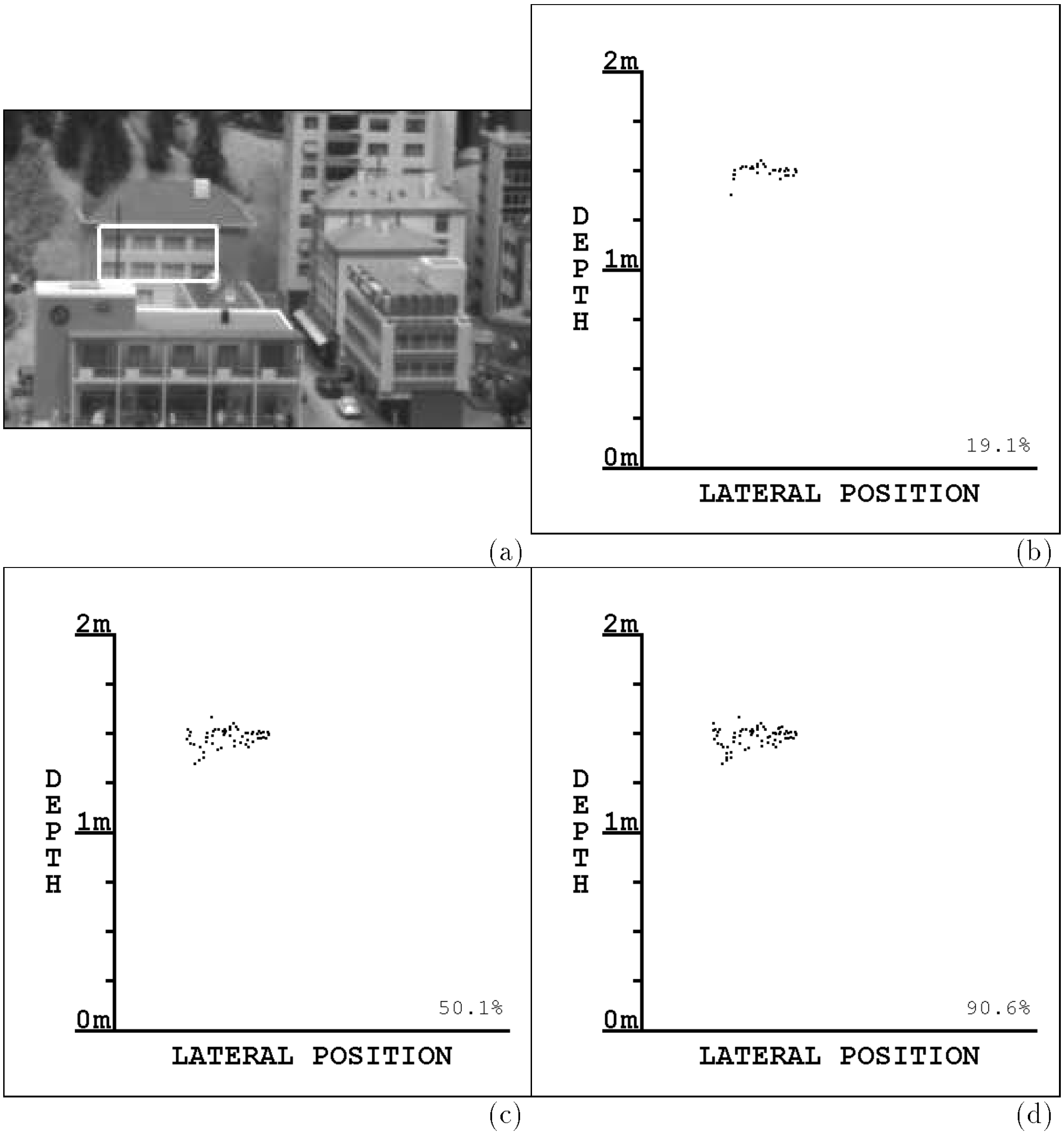


Figure 7-4: Depth Estimates for CMU Sequence.

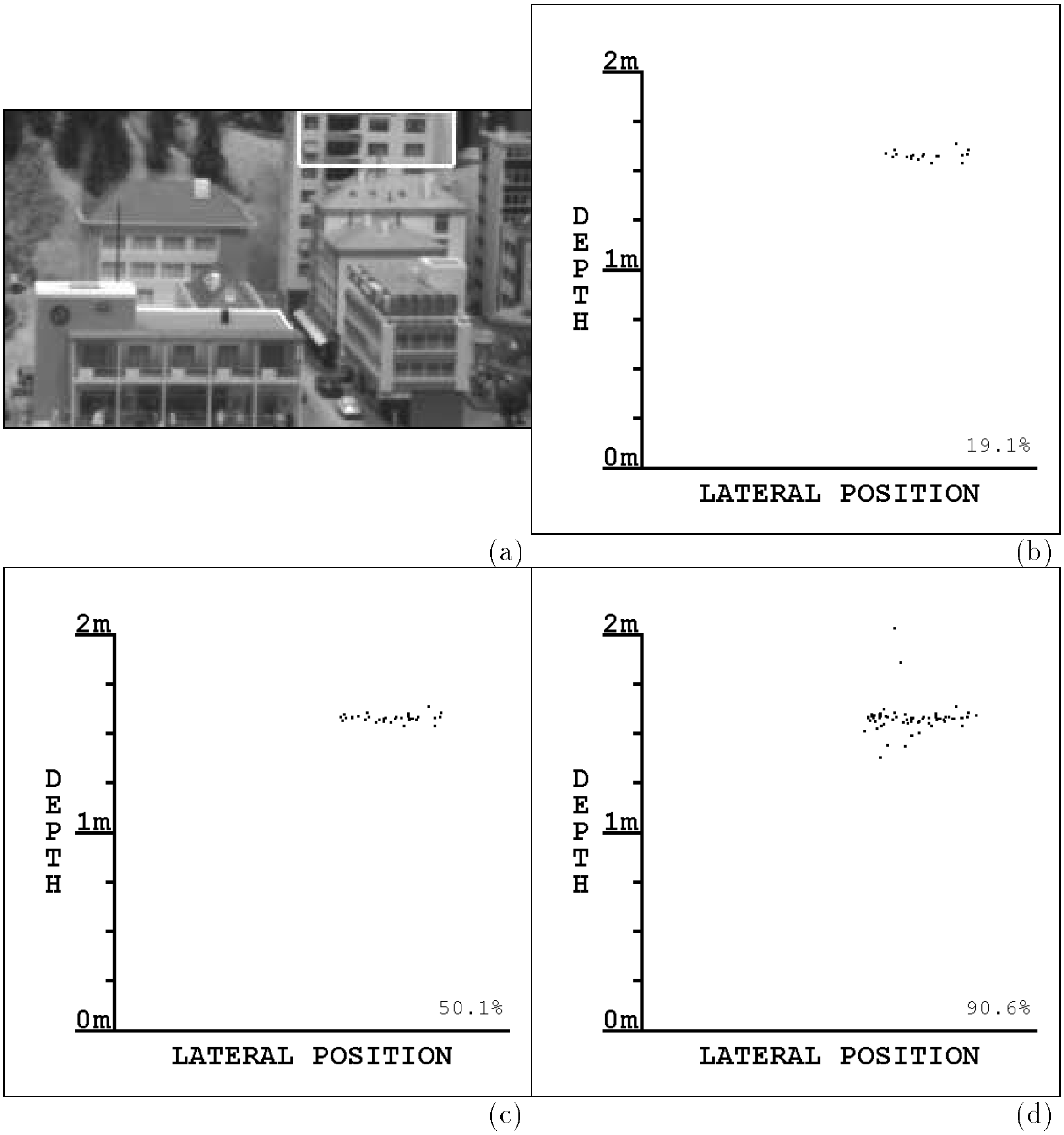


Figure 7-5: Depth Estimates for CMU Sequence.

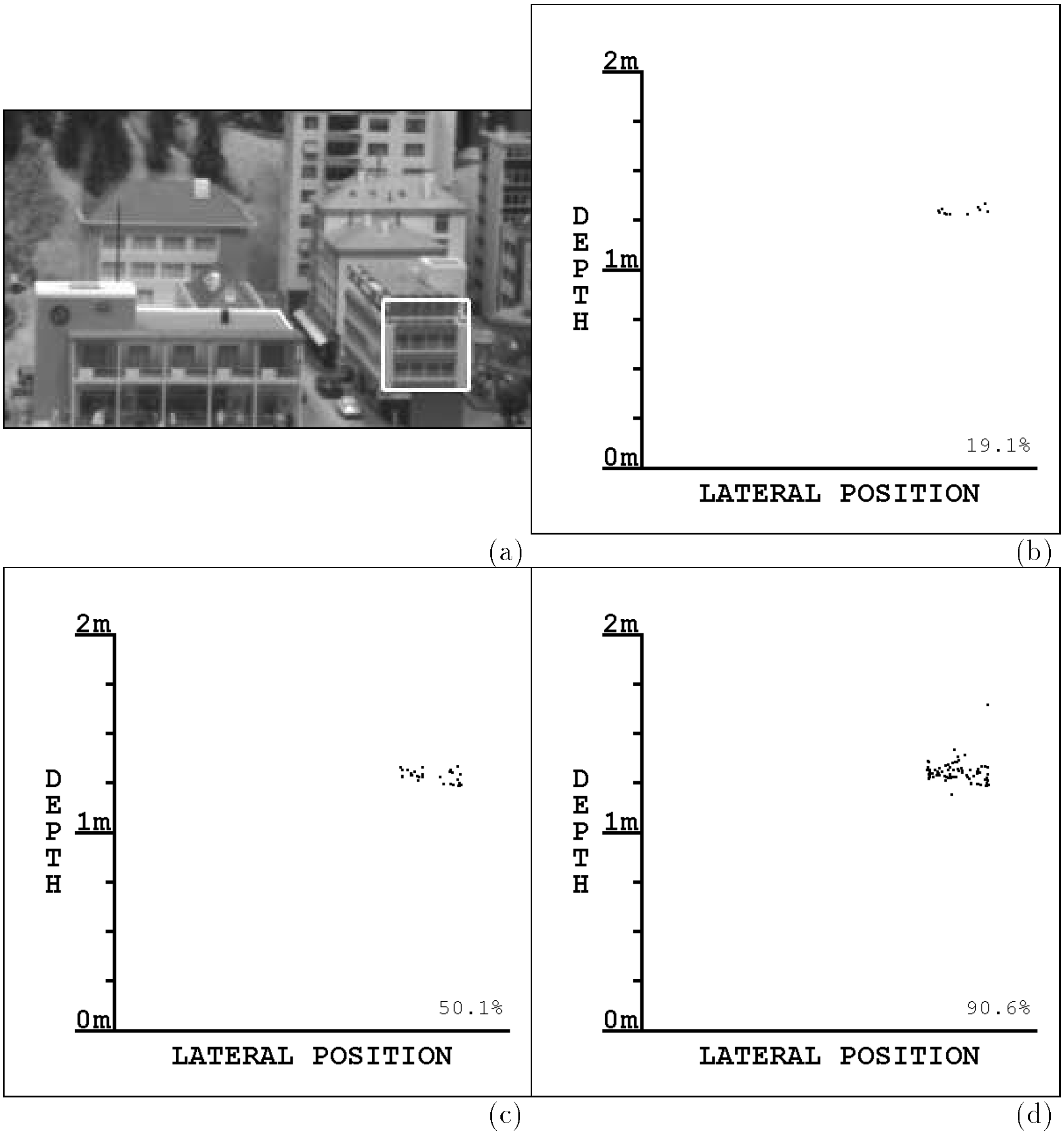


Figure 7-6: Depth Estimates for CMU Sequence.

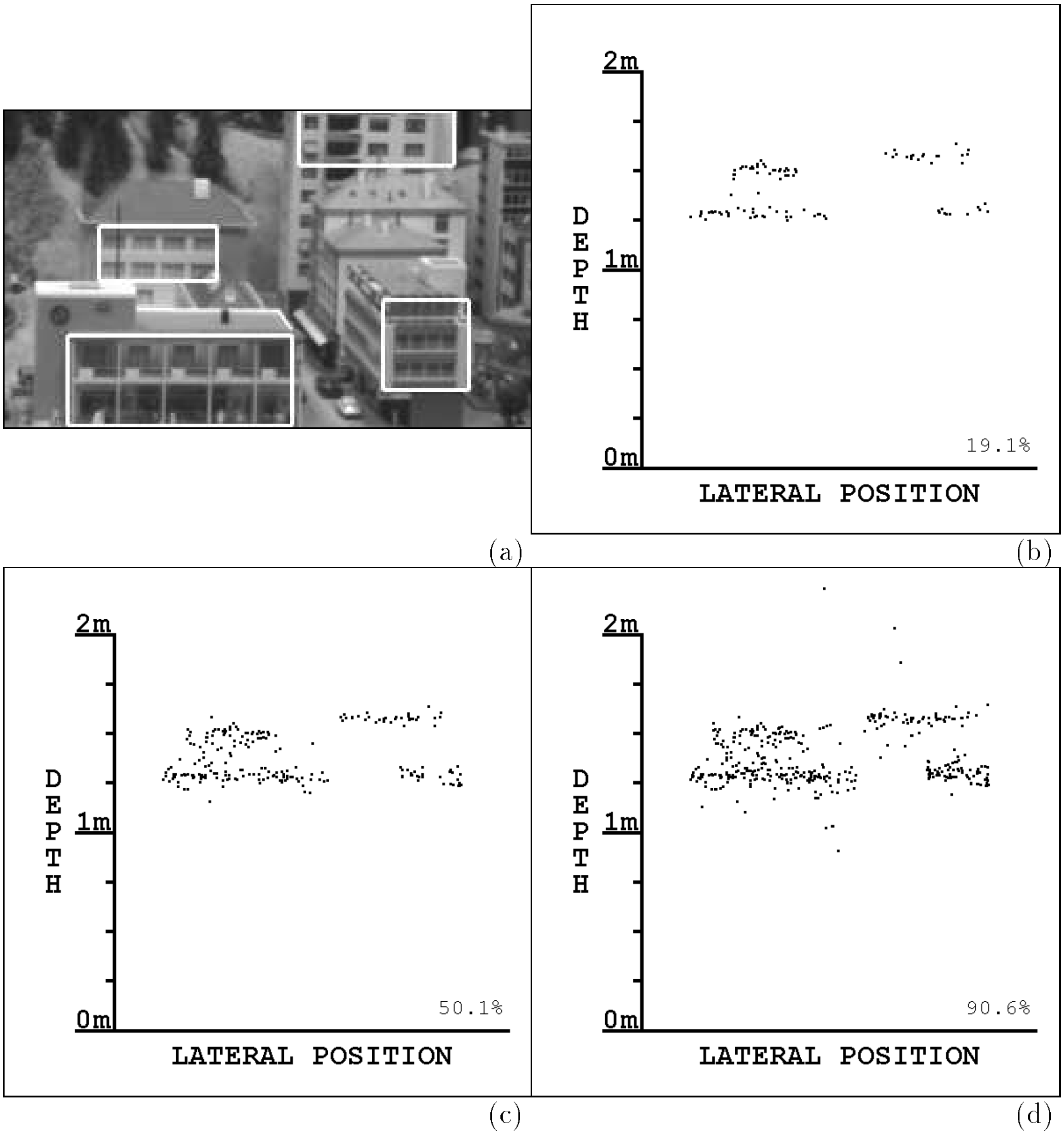
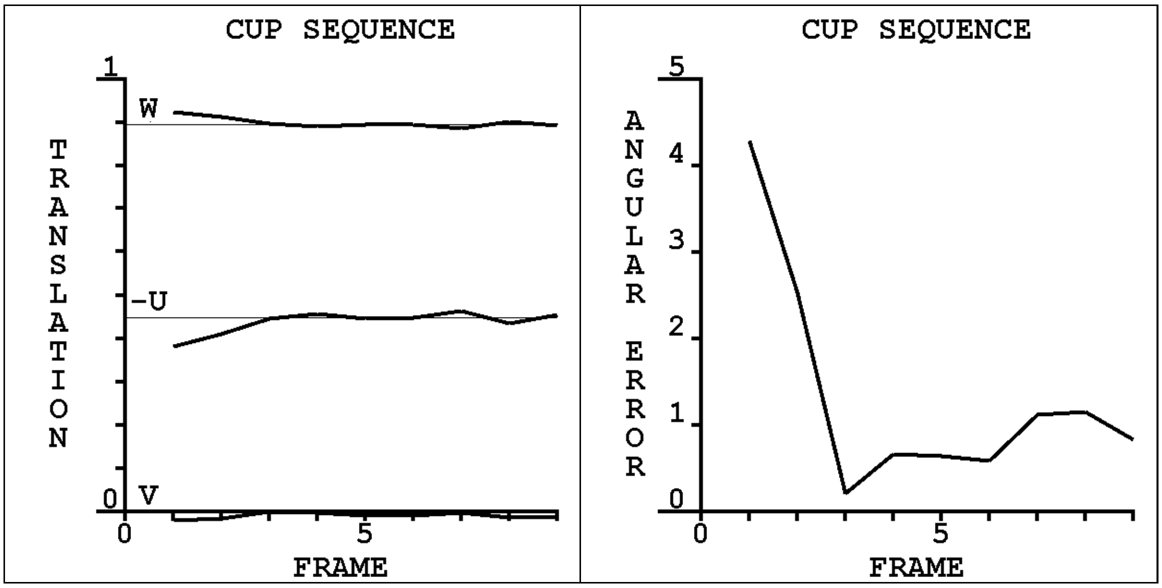


Figure 7-7: Depth Estimates for CMU Sequence.

Figure 7-8: First Frame of the Cup Sequence.



(a) (b)

Figure 7-9: Translation Vector Estimates. In (a), the normalized components of the estimated translation vector are plotted versus the frame number of the cup sequence. The correct values, $(-.447, 0., .894)^{T}$, are superimposed on the plot. (The value of U is negated for convenience.) In (b), the angular error between the estimated translation vector and the correct translation vector is plotted versus the frame number of the sequence.

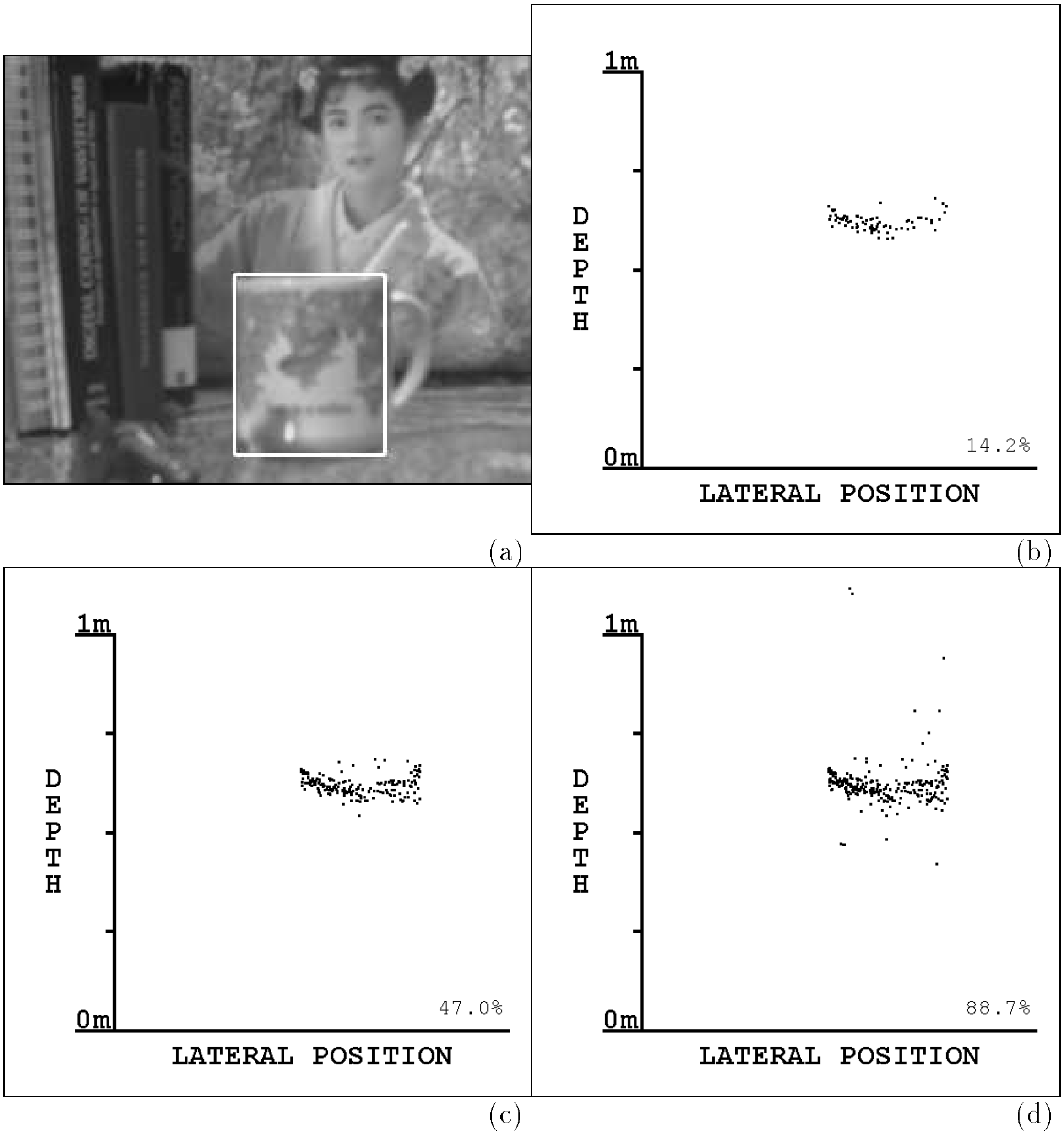


Figure 7-10: Depth Estimates for Cup Sequence.

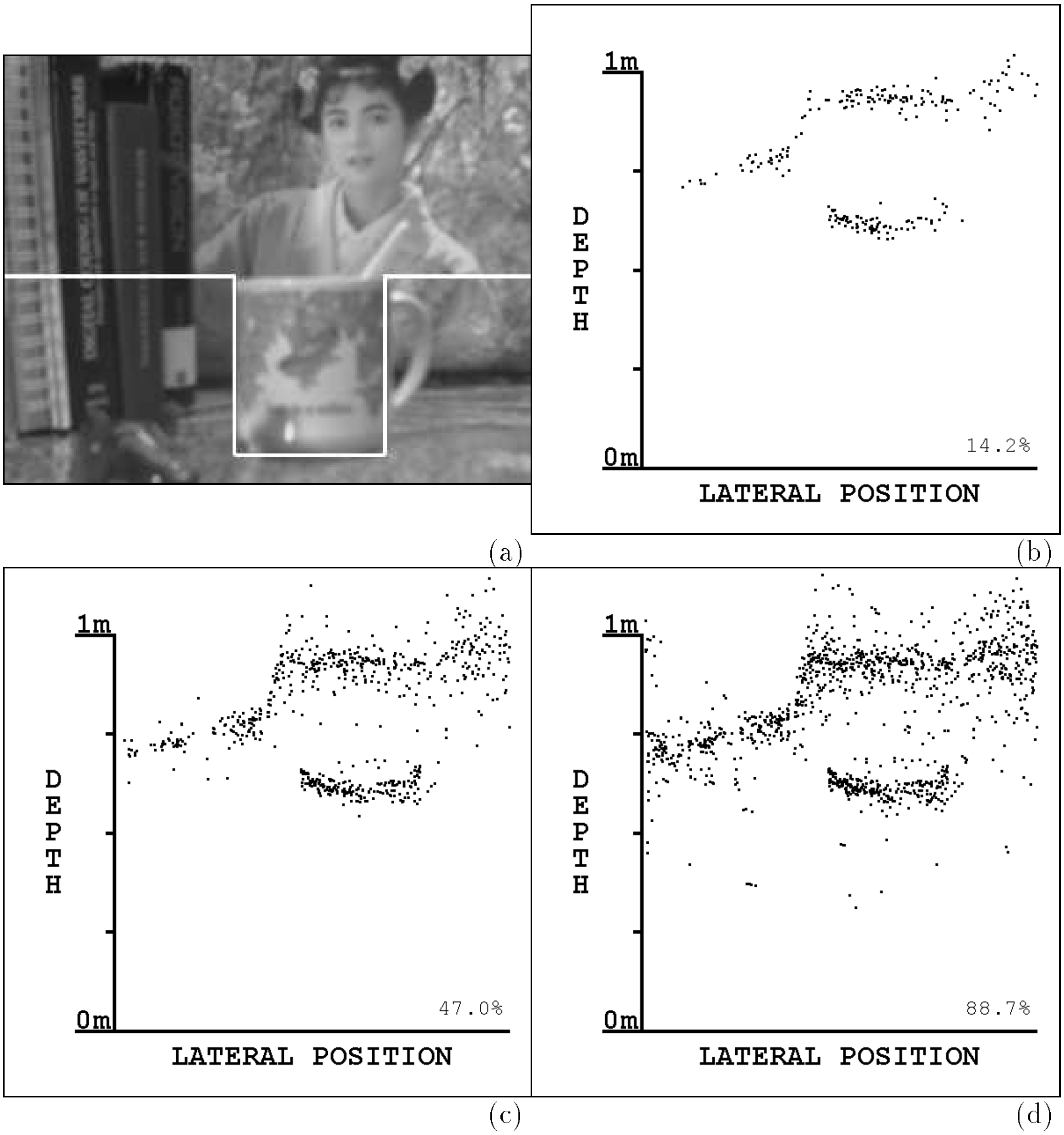


Figure 7-11: Depth Estimates for Cup Sequence. The depth estimates for features located above the line in the image are depicted.

7.5 Discussion

The weighted least square-error algorithm provides estimates of the direction of the translation vector and the relative depth of each feature. The weighting scheme allows the algorithm to exploit *a priori* information pertaining to the accuracy of the flow measurement. Measurements believed to be more accurate are given greater weight than those believed to be less accurate.

The variance measure provided by the recursive flow estimator is a natural basis for these arbitrary weights. The variance is a direct measure of the reliability of the flow estimate. As the flow estimate of a particular feature improves over time, it is given increasing weight in the translation vector estimate.

More specifically, the weight given to each flow estimate is the reciprocal of the variance reported by the recursive estimator. This choice of weights is suggested by a number of results in stochastic theory. Consider, for example, a sequence of independent random variables that have a constant, unknown mean and non-constant variances. The linear least square-error estimate of the mean is given by a weighted sum of the random variables. In order to obtain the optimal solution, each weight is chosen such that it is proportional to the reciprocal of the variance of the respective random variable.

Of course, there are alternate choices of the weights for the translation vector estimator. Other types of information may be incorporated into the estimation process via the weights. For example, a function of the values of the Laplacian of Gaussian signal within each simple region may be used. In this case, features corresponding to regions with higher contrast would have greater influence than features corresponding to lower contrast. This type of weighting would play the same role as the thresholding stage in edge detection. The effects of weak features, in this sense, would be reduced. However, in contrast to edge detection, the weaker features would not be eliminated entirely.

For the results presented in Section 7.4, the recursive estimator plays a significant role in the performance of the system. As illustrated in Figures 7-2 and 7-9, the translation vector estimate improves rapidly over time then remains within a steady-state error bound. The temporal improvement in the estimate is facilitated by two effects. First, the flow estimates are improved over time by the recursive estimator. The improved quality of the input data for translation vector estimator leads directly to an improvement in the output. The second factor for the temporal improvement is the heavier weighting of the more reliable features. As we discussed in Section 6.4, the variance measure is equivalently a measure of the persistence and stability of the feature. The weighting function reduces the influence of spurious features.

The variance also provides a measure of the accuracy of the depth estimate. Since the depth estimate of a particular feature is a function of the flow estimate of the feature, the accuracy of the depth estimate is related to the accuracy of the flow estimate. A more reliable flow estimate yields a more reliable depth estimate. Therefore, the variance of the flow is an indirect measure of the reliability of the depth estimate.

As a result, the algorithm is able to distinguish the reliable depth estimates from those that are less reliable. The algorithm provides a relatively sparse depth map that is extremely reliable. The algorithm also provides additional depth estimates with diminishing reliability and accuracy. There is an explicit tradeoff between the density of the depth map and the average accuracy of the depth map. The algorithm quantifies the tradeoff with the variance measure.

This tradeoff is illustrated in the results of Section 7.4. In each of the figures depicting depth estimates, the accuracy of the depth is strongly correlated with the variance measure. The best depth map in each figure corresponds to the one with the lowest variance values (subfigure (b)). The depth estimates corresponding to the highest variance values are the least reliable (subfigure (d)).

7.6 Summary

In this chapter, we consider a weighted least-square error approach for estimating the translation vector and the relative depth from the flow estimates of simple region features. The weighting scheme exploits the variance measure of the optical flow estimate for each token. In addition, the variance provides a measure of the reliability of each depth estimate.

Chapter 8

Discussion

The simple region feature extraction paradigm is a novel approach for abstracting information from an image or sequence of images. Simple region features have significant advantages over existing feature extraction paradigms. In this chapter, we explore the merit of the paradigm and consider implications for future research.

8.1 Discussion

Because edge detection is the most popular feature extraction paradigm, we compare simple region features to edges. The simple region feature extraction paradigm has a number of fundamental advantages over edge detection. Most of the advantages result from the more abstract nature of simple region features compared to edges. Each simple region contains more information than a single edge. Another source of advantage is that simple regions are "extracted" from the image rather than "detected", as we discuss next.

One important difference between simple region features and edges is that there is no inherent threshold present in the simple region feature extraction process. The term "detection" implies that a decision is made with regard to the presence or absence of an edge at each point in the image. The decision is made by applying a threshold to the output of a filter in the neighborhood of the edge. If the edge does not meet the threshold, it is discarded. As a consequence, edges are sensitive to small changes in the brightness, particularly when the output of the filter is close to the value of the threshold. When considering edges from a sequence of images, the edges tend to disappear and reappear as the sequence evolves.

In contrast, simple region features are not sensitive to any particular threshold because no threshold exists. All of the features that are obtained from the extraction process are retained in the representation. Rather than eliminating the features based on an arbitrary threshold, subsequent processes may give the features different weights based on a criterion that is appropriate to the task.

For example, in Chapter 7, we use a weighting scheme that is based on the variance measure provided by the recursive flow estimator of Chapter 6. The variable weighting scheme is preferable to a thresholding mechanism because of two considerations. First, it allows the information associated with all the features to be considered. In contrast, employing a threshold implies that only a subset of the features are considered. Second, greater consideration is given to the information that is deemed more reliable.

Alternative weighting schemes are possible for the simple region features. For example, the value of the Laplacian of Gaussian filter in the area occupied by a particular feature could also serve as the basis for a weighting scheme. In this context, the weights would incorporate the same information that is used by an edge detection threshold without eliminating any of the features from consideration. As a result, the simple region features are capable of providing a more dense representation than edges.

Another difference is that simple regions possess more abstract attributes than do edges. An edge consists of a location and possibly a crude estimate of the orientation. In contrast, the simple region features possess, for example, location, area, and orientation. In short, simple regions have shape and size; edges do not.

Throughout the development and demonstration of the feature extraction paradigm, we exploit the abstract properties of simple regions. In Chapter 6, for example, the geometric properties and the shape of the simple regions are used to determine the correspondence of features between frames. A similarity measure based on the difference of the area of corresponding features enhances the ability to track the features in a sequence of images. The shape of the features is exploited to reduce the computational complexity of the correspondence algorithm significantly.

In addition, the abstract properties facilitate the computation of a persistence measure of the features. The persistence measure is based on the stability of the shape of corresponding features and their respective neighbors. The persistence measure enables subsequent algorithms to give greater weight to the more reliable features. For example, in Chapter 7 we exploit the persistence measure to improve the egomotion estimates and to differentiate the quality of the depth estimates.

In contrast, edges do not have shape and size attributes. There is little information to use as the basis for a similarity measure between two edges. One might base such a measure on the orientation of the edge and the brightness in the neighborhood of the edge. However, such a measure is likely to be unreliable. Similarly, determining the persistence and stability of an edge over a sequence of images would also be difficult. There is little or no basis to measure the stability of the track over time.

The results presented in Chapter 7 demonstrate that the simple region feature extraction paradigm is a viable approach. For the test sequences considered, the motion estimation algorithm provides the translation direction accurate to within one or two degrees. Such accuracy has been reported by a variety of authors to be sufficient for most navigational tasks (for a review, see Hildreth[33]).

Furthermore, the algorithm provides reasonable depth estimates for a large percentage of the features. A small number of outliers are present in the output due to errors in the correspondence algorithm. However, as the correspondence algorithm is refined, the number of outliers is likely to be reduced. In addition, the current implementation does not include a surface model or spatial smoothing of the depth estimates. The incorporation of a surface reconstruction technique into the algorithm is likely to eliminate many of the outliers and to improve the overall accuracy of the depth estimates.

We have considered the utility of simple region features almost exclusively in the context of the passive navigation problem. However, the simple region feature extraction paradigm is more than a method for estimating egomotion and depth. The paradigm is applicable to a variety of machine vision problems. In the next section, we consider simple region features in a broader context as the foundation for a novel early vision paradigm.

8.2 Future Directions

Simple regions are applicable to a variety of machine vision problems. Potentially, simple regions will provide improvements to each of the problems independently. However, the real advantage of using the paradigm for a spectrum of problems is that the resulting algorithms will be built on the same foundation. As a result, the combination of information from individual components of a vision system will become more straightforward.

In addition to the passive navigation problem, the simple region feature paradigm is directly applicable to stereopsis and object recognition. Many of the solutions that have been proposed for these problems are based on manipulation of features. We briefly consider each of these problems in the context of simple region features.

Stereopsis is the determination of the depth and shape of objects by using two (or more) distinct views of the same environment. Many stereo algorithms are based on the determination of the correspondence mapping of features from two images. From the correspondence mapping, the algorithms determine the disparity: the displacement between corresponding features in the respective images. Once the disparity is known, the depth is recovered from a well-known geometric technique called triangulation. For a more detailed discussion of stereo vision, see, for example, Grimson[25], [26], Horn[41], or Ballard & Brown[5].

Passive navigation and stereopsis are dual problems. Both involve combining information from two or more images of the same environment. As a result, the issues related to stereopsis are very similar to those of motion. For example, both require the determination of correspondence of features from two similar images. The geometric relationships involved in computing the depth are also similar in each case. This suggests that simple region features are applicable to stereo as well as passive navigation.

In practice, one difference between motion and stereo is that the baseline, the displacement of the camera(s), is typically greater in the case of stereo. In addition, the baseline is usually known, at least approximately, for stereo systems. As a result, there is an enormous constraint placed on the relative positions of corresponding features. The corresponding points from a stereo pair must both lie on the same *epipolar line*. Each epipolar line is the intersection of a plane that contains the centers of projection of both cameras and the image plane. For a more detailed discussion, see[25], [41], or [5].

Due to the subtle differences in the two problems, the demands placed on the correspondence algorithm differ between stereo and motion systems. Because of the larger baseline for stereo systems, the disparity or displacement between features is typically larger. However, due to the epipolar constraint, the displacement of corresponding features is in a known direction.

These observations suggest that by modifying the correspondence algorithm, simple region features may be used as the basis for a stereo algorithm. Such a stereo correspondence algorithm would prefer correspondences that are consistent with the epipolar constraint. It would also be important to exploit multiple resolution processing due to the increased displacement of the features. Once a suitable correspondence algorithm is specified, the depth may be determined using well-known geometric principles.

Object recognition is another problem that may be addressed using simple region features. Currently, most object recognition paradigms are based on the idea of matching features extracted from an image to features from a model that represents an object. By choosing the appropriate model from a data base, the object is identified (see, for example, Grimson[27]). Often, the object recognition algorithm uses Canny edges as the feature that is matched. However, there is no fundamental reason that precludes the development of an algorithm that uses simple region features instead.

The same general issues that confront current object recognition systems would pertain to a system based on simple region features. The system would be required to generate the hypothetical pose of a candidate object based on the configuration of a subset of features in the image. The system would then verify or reject the hypothesis by considering additional features. The use of techniques to reduce the computational burden, such as grouping and indexing, would also be necessary (for example, Jacobs[44], Clemens & Jacobs[21], or Lowe[48]).

Of course, each of these issues must be reconsidered in the context of simple region features. Most notably, the models of objects in the data base would be much different than the models in systems that use Canny edges. The models should be constructed such that they efficiently represent the information necessary for matching the data to the model. Due to the different nature of simple region features versus Canny edges, it would be desirable to customize the model representation to simple region features. Similarly, the algorithm to match the features to the model would also need to be customized.

In the cases of passive navigation, stereopsis, and object recognition, systems based on simple region features are likely to obtain performance that is comparable (if not improved) with respect to existing methods. The primary advantage of applying simple regions to all of these problems is that combining information from the individual modules will become more straightforward. By using the same internal representation in each module, communication among the modules is enhanced. We call this idea *module integration via shared representation.*

Consider two modules that use different internal representations. One module is a stereo algorithm that uses zero-crossings of the Laplacian of Gaussian filter as the basis for correspondence; the Grimson[27] approach, for example. The other module is an object recognition algorithm that uses Canny edges. Suppose that for a particular hypothetical pose of an object, the model predicts a substantial difference in depth between two of the Canny edges. Comparing the predicted depth difference with the depth estimated from the stereo would provide significant information about the validity of the hypothesis. However, it would be non-trivial to find zero-crossings from the stereo representation that correspond to the Canny edges in question. Exploiting information from the stereo algorithm in this case would be awkward at best.

In contrast, consider a stereo module and an object recognition module that both use simple region features. In this case, there is no problem obtaining a correspondence between the internal representations of the modules because the representations are identical. If the object recognition system requires depth information pertaining to a particular feature, the stereo algorithm provides the depth information for that specific feature.

Consider also the advantage of a passive navigation module and an object recognition module that use the same internal representation. Suppose that an object has been identified in a particular frame of an image sequence. When the next frame is processed it is desirable to exploit the correspondence mapping obtained by the motion module. Doing so would facilitate an efficient update of the position of the object of interest in subsequent images.

More specifically, the correspondence mapping of the motion module and the mapping from the model to the features in the previous frame together imply a mapping between the model and features in the next frame. Of course, it would be necessary to verify the new mapping and correct possible errors. However, this is substantially more efficient than determining the model to feature mapping in the next frame independent of the previous frame mapping. This process would be considerably more manageable if the passive navigation module and the object recognition module use the same internal representation.

Ultimately, a primary advantage of the simple region feature extraction paradigm is that it is useful for a variety of problems. While the paradigm may or may not be the best approach for a particular problem, the advantage of using the same features for each of the modules is enormous. Communication of information among the modules becomes much more tractable in this case. In this sense, the simple region features serve as the foundation of a novel early vision paradigm.

Chapter 9

Conclusion

Throughout this thesis, we stress the importance information representation. In Part I, an analytical representation for contours facilitates the computation of a novel scale-space for contours and the medial axis skeleton. In Part II, a novel feature abstraction provides a rich description of the image.

The analytical contour representation yields a number of benefits. The representation provides geometric properties, such as the position, orientation, and curvature, explicitly. The representation facilitates a novel approach to smoothing the contour; the approach depends heavily on the explicit representation of the curvature.

The contour representation implies a complementary analytical representation for the medial axis skeleton. As a result of the representation strategy, the skeleton is determined uniquely from the contour. Furthermore, the skeleton computation is robust against perturbations in the bounding contour and the input data.

The combination of the beneficial properties of the contour and skeleton representations facilitates the computation of the complexity scale-space. For example, the explicit representation of the contour curvature is essential to the scale-space computation. Similarly, the unique mapping of the contour to the skeleton and the stability of the skeleton with respect to contour are crucial to the reliable computation of the scale-space.

Similarly, the advantages of simple regions over other features stem directly from the representational strategy. The most important distinction between simple regions and other features is the abstract nature of the simple regions. The geometric attributes of the simple regions provide substantial advantages in a variety of contexts.

The motion estimation algorithm exploits the geometric attributes of simple regions in several ways. The correspondence algorithm uses a simple measure of the similarity of the features. Once the correspondence mapping is determined, a measure of the reliability of the optical flow estimate is determined based on the shapes of the corresponding features. The algorithm to determine the direction of translation of the camera exploits the measure by giving greater weight to the more reliable flow estimates. Furthermore, the reliability measure also applies to the relevant depth estimates.

Each of these capabilities is made possible by the geometric attributes of the simple region features. Overall, these properties are an example of the benefit of an appropriate representation strategy. Information captured in the representation of the most primitive stage of the system leads to enhanced cabilities in the subsequent stages of the system. More specifically, the geometric attributes of the features ultimately lead to the ability to discern the reliability of depth estimates.

In the thesis, we also argue that existing information representation strategies are insufficient for the ultimate goal of a general vision system. Edges, which are currently the most popular features, constitute a sparse and unreliable description of the image. Much of the pertinent information in the image is discarded in the edge detection process.

The simple region feature extraction paradigm has a number of advantages over edge detection. The simple region features are more reliable and provide a more dense representation of the image information. Simple regions possess more abstract properties than edges. The additional information encoded in each feature facilitates enhanced capabilities in the subsequent processing stages.

The ultimate test of the paradigm will be its applicability and effectiveness for a variety of vision subproblems. We have demonstrated that simple region features are useful in the context of passive navigation. Furthermore, we argue that simple regions are applicable to other vision modules. If future efforts to apply simple region features to a variety of modules are successful, the synthesis of a consistent representational framework for these modalities will be made possible. Combining information among the modules will become more straightforward. And, as as result, the overall system will provide more complete and reliable information about the environment.

The representation of information is a critical issue in the design of machine vision systems. The representation strategy at the most primitive stages has enormous impact on the overall capabilities of the system. Given the shortcomings of current approaches, it is essential to consider novel methods. It is necessary to improve the representational capabilities at all levels within the system. To achieve this goal, we must continually reevaluate the fundamental assumptions underlying the design and development of vision systems.

Bibliography

[1] J. K. Aggarwal and N. Nandhakumar. On the computation of motion from sequences of images—a review. *Proceedings of the IEEE*, 76(8):917–935, 1988.

[2] B. D. O. Anderson and J. B. Moore. *Optimal Filtering*. Prentice-Hall, Englewood Cliffs, NJ, 1979.

[3] H. Asada and M. Brady. The curvature primal sketch. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(1):2–14, January 1986.

[4] F. Attneave. Some aspects of visual perception. *Psychology Review*, 61:183–193, 1954.

[5] D. H. Ballard and C. M. Brown. *Computer Vision*. Prentice-Hall, Englewood Cliffs, NJ, 1982.

[6] M. Bertero, T. Poggio, and V. Torre. Ill-posed problems in early vision. AI Memo 924, MIT Artificial Intelligence Laboratory, May 1987.

[7] V. Berzins. Accuracy of Laplacian edge detectors. *Computer Vision, Graphics, and Image Processing*, 27:195–210, 1984.

[8] H. Blum. A transformation for extracting new descriptions of shape. In *Symposium on Models for the Perception of Speech and Visual Form*. MIT Press, Cambridge, MA, 1964.

[9] T. J. Broida and R. Chellappa. Kinematics and structure of a rigid object from a sequence of noisy images. In *Proceedings Workshop on Motion Representation and Analysis*. IEEE Computer Society, 1986.

[10] A. Bruss and B. K. P. Horn. Passive navigation. *Computer Vision, Graphics, and Image Processing*, 21(1):3–20, 1983.

[11] W. Burger and B. Bhanu. Estimating 3-D egomotion from persepective image sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-12:1040–1058, 1990.

[12] P. J. Burt and E. H. Adelson. The Laplacian pyramid as a compact image code. *IEEE Trans. on Commm.*, COM-31(4):532–540, April 1983.

[13] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986.

[14] R. D. Chaney. Analytical representation of contours. In *Proceedings on Intelligent Robots and Computer Vision X: Algorithms and Techniques*, Boston, MA, November 1991.

[15] R. D. Chaney. Analytical representation of contours. AI Memo 1392, MIT Artificial Intelligence Laboratory, October 1992.

[16] R. D. Chaney. Computation of the medial axis skeleton at multiple complexities. In *Proceedings on Intelligent Robots and Computer Vision XI: Algorithms, Techniques, and Active Vision*, Boston, MA, November 1992.

[17] R. D. Chaney. Complexity as a scale-space for the medial axis transform. AI Memo 1397, MIT Artificial Intelligence Laboratory, January 1993.

[18] P. C. Chen and T. Pavlidis. Segmentation by texture using a co-occurrence matrix and a split-and-merge algorithm. *Computer Graphics and Image Processing*, 10(2):172–182, June 1979.

[19] R. T. Chin, H. K. Wan, D. L. Stover, and R. D. Iversion. A one-pass thinning algorithm and its parallel implementation. *Computer Vision, Graphics, and Image Processing*, 40(1):30–40, 1987.

[20] D. T. Clemens. The recognition of two-dimensional modeled objects in images. Master's thesis, Massachusetts Institute of Technology, Department of Electical Engineering & Computer Science, 1986.

[21] D. T. Clemens and D. W. Jacobs. Space and time bounds on indexing 3-d models from 2-d images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):1007–1017, October 1991.

[22] L. S. Davis. A survey of edge detection techniques. *Compter Graphics and Image Processing*, 4(3):248–270, September 1975.

[23] A. R. Dill, M. D. Levine, and P. B. Nobel. Multiple resolution skeletons. *Computer Vision, Graphics, and Image Processing*, 40(1):30–40, 1987.

[24] O. D. Faugeras and S. Maybank. Motion from point matches: multiplicity of solutions. *International Journal of Computer Vision*, 4(3):225–246, June 1990.

[25] W. E. L. Grimson. *From Images to Surfaces.* MIT Press, Cambridge, MA, 1981.

[26] W. E. L. Grimson. Computational experiments with a feature based stereo algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-7(1):17–34, January 1985.

[27] W. E. L. Grimson. *Object Recognition by Computer: The Role of Geometric Constraints*. MIT Press, Cambridge, MA, 1990.

[28] W. E. L. Grimson. On the recognition of curved objects in two dimensions. In H. Nasr, editor, *Automatic Object Recognition*. SPIE Optical Engineering Press, 1991.

[29] R. M. Haralick. Zero-crossings of second directional derivative edge operator. In *SPIE Proceedings on Robot Vision*, Arlington, VA, 1982.

[30] J. Heel. *Temporal Surface Reconstruction*. PhD thesis, Massachusetts Institute of Technology, Department of Electical Engineering & Computer Science, 1991.

[31] E. C. Hildreth. The detection of intensity changes by computer and biological vision systems. *Computer Vision, Graphics, and Image Processing*, 22:1–27, 1983.

[32] E. C. Hildreth. Edge detection. In S. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 257–267. John Wiley & Sons, New York, 1987.

[33] E. C. Hildreth. Recovering heading for visually-guided navigation. AI Memo 1297, MIT Artificial Intelligence Laboratory, June 1991.

[34] E. C. Hildreth and S. Ullman. The computational study of vision. AI Memo 1038, MIT Artificial Intelligence Laboratory, April 198.

[35] S. B. Ho and C. R. Dyer. Comparison of thinning algorithms on a parallel processor. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(4):512–520, 1986.

[36] D. D. Hoffman and W. A. Richards. Parts of recognition. *Cognition*, pages 65–96, 1984.

[37] D. D. Hoffman and W. A. Richards. Representing smooth plane curves for recognition: Implications for figure-ground reversal. In W. A. Richards, editor, *Natural Computation*, pages 76–82. MIT Press, Cambridge, MA, 1988.

[38] B. K. P. Horn. The Binford-Horn LINEFINDER. AI Memo 285, MIT Artificial Intelligence Laboratory, December 1973.

[39] B. K. P. Horn. The curve of least energy. *ACM Transactions on Mathematical Software*, 9(4):441–460, December 1982.

[40] B. K. P. Horn. Extended Gaussian images. *Proceedings of the IEEE*, 72(12):1671–1686, December 1984.

[41] B. K. P. Horn. *Robot Vision.* MIT Press, Cambridge, MA, 1986.

[42] B. K. P. Horn and E. J. Weldon Jr. Filtering closed curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(5):665–668, September 1986.

[43] S. L. Horowitz and T. Pavlidis. Picture segmentation by a tree traversal algorithm. *Journal of the ACM*, 23(4):368–388, April 1976.

[44] D. W. Jacobs. *Recognizing 2D Objects Using 3D Images.* PhD thesis, Massachusetts Institute of Technology, Department of Electical Engineering & Computer Science, 1992.

[45] J. J. Koenderink. *Solid Shape.* MIT Press, Cambridge, MA, 1990.

[46] Y. G. Leclerc. Constructing simple stable descriptions for image partitioning. *International Journal of Computer Vision*, 3(1):73–102, May 1989.

[47] F. Leymarie and M. D. Levine. Simulating the grassfire transform using an active contour model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-14(1):56–75, January 1992.

[48] D. G. Lowe. *Perceptual Organization and Visual Recognition.* Kluwer Academic Publishers, 1985.

[49] D. G. Lowe. Organization of smooth image curves at multiple scales. *International Journal of Computer Vision*, 3(2):119–130, June 1989.

[50] D. G. Luenberger. *Introduction to Dynamic Systems: Theory, Models, & Applications.* John Wiley & Sons, New York, 1979.

[51] W. H. H. Lunscher. The asymptotic optimal frequency domain filter for edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:678–680, 1983.

[52] D. Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information.* W. H. Freeman & Co., San Francisco, 1982.

[53] D. H. Marr and E. C. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London B*, 204:301–328, 1980.

[54] M. P. Martínez-Peréz, J. Jiménez, and J. Navalón. A thining algorithm based on contours. *Computer Vision, Graphics, and Image Processing*, 39(2):186–201, August 1987.

[55] J. W. McKee and J. K. Aggarwal. Computer recognition of partial views of curved objects. *IEEE Transactions on Computers*, 26(8):790–800, 1977.

[56] D. J. S. Michael. *Exploiting Continuity-in-Time in Motion Vision.* PhD thesis, Massachusetts Institute of Technology, Department of Nuclear Engineering, 1992.

[57] F. Mokhtarian and A. K. Mackworth. Scale-based description and recognition of planar curves and two-dimensional shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(1):34–43, January 1986.

[58] M. Okutomi and T. Kanade. A multiple-baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-15(4):353–363, April 1993.

[59] J. Oliensis. Local reproducible smoothing without shrinkage. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-15(3):307–312, March 1993.

[60] A. Papoulis. *Probability, Random Variables, and Stochastic Processes.* McGraw-Hill, New York, 2nd edition, 1984.

[61] T. Pavlidis. Algorithms for shape analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4:301–312, July 1980.

[62] T. Pavlidis and S. L. Horowitz. Segmentation of plane curves. *IEEE Transactions on Computers*, 23(8):860–870, August 1974.

[63] W. A. Perkins. Simplified model-based part locator. *IEEE Transactions on Computers*, 27(2):126–143, February 1978.

[64] W. Richards and A. Jepson. What makes a good feature? A.I. Memo 1356, MIT Artificial Intellegence Lab, April 1992.

[65] W. A. Richards, B. Dawson, and D. Whittington. Encoding contour shape by curvature extrema. In W. A. Richards, editor, *Natural Computation*, pages 83–98. MIT Press, Cambridge, MA, 1988.

[66] J. Rissan. Minimum-description-length principle. In *Encyclopedia of Statistical Sciences*, volume 5, pages 523–527. John Wiley & Sons, New York, 1987.

[67] L. G. Roberts. Machine perception of three-dimensional solids. In J. T. Tippet et al., editor, *Optical and Electro-Optical Information Processing*, pages 159–197. MIT Press, Cambridge, MA, 1965.

[68] A. Rosenfeld and A. C. Kak. *Digital Picture Processing.* Academic Press, New York, 2nd edition, 1982.

[69] K. F. Shanmugam, F. M. Dickey, and J. A. Green. An optimal frequence domain filter for edge detection in digital pictures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1:37–49, 1979.

[70] H. Shariat and K. Price. Motion estimation with more than two frames. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(5):417–434, May 1990.

[71] G. B. Thomas, Jr. and R. L. Finney. *Calculus and Analytical Geometry.* Addison Wesley Publishing Co., Reading, CA, 5th edition, 1980.

[72] V. Torre and T. Poggio. On edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(2):147–163, March 1986.

[73] H. L. Van Trees. *Detection, Estimation, and Modulation Theory Part I.* John Wiley & Sons, New York, 1968.

[74] R. Y. Tsai. Multiframe image point matching and 3-D surface reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(2), 1983.

[75] J. L. Turney, T. T. Mudge, and R. A. Volz. Recognizing partially occluded parts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(4):410–421, 1985.

[76] N. Ueda and S. Suzuki. Learning visual models from shape contours using multiscale convex/concave structure matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-15(4):337–352, April 1993.

[77] S. Ullman. *The interpretation of visual motion.* MIT Press, Cambridge, MA, 1979.

[78] P. Wallach. Garbage in, garbage out: Simple geometry brings supercomputers to their knees. *Scientific American*, page 126, December 1990.

[79] J. Weng, N. Ahuja, and T. S. Huang. Matching two perspective views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-14(8):806–825, August 1992.

[80] A. P. Witkin. Scale space filtering. In *Proc. 7th International Conference on Artificial Intelligence*, pages 1019–1021, Karlsruhe, 1983.

[81] Y. Xia. Skeletonization via the realization of the fire front's propagation and extinction in binary shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(10):1076–1086, October 1989.

[82] G. S. Young and R. Chellappa. 3-d motion estimation using a sequence of noisy stereo images. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pages 710–716, 1988.