MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A.I. Memo No. 1392 October 1992

Analytical Representation of Contours

Ronald D. Chaney

Abstract

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. In this paper, we present an analytical representation for contours. The contour is represented by a set of pairwise tangent circular arcs. The advantage of such an approach is that mathematical properties such as orientation and curvature are explicitly represented. We introduce a smoothing criterion for the contour that optimizes the tradeoff between the complexity of the contour and proximity of the data points. The complexity measure is the number of extrema of curvature present in the contour. The smoothing criterion leads us to a true scale-space for contours. We describe the computation of the contour representation as well as the computation of relevant properties of the contour. We consider the potential application of the representation, the smoothing paradigm, and the scale-space to contour interpretation and recognition.

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.

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[2] 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). 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[31]. For 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 discreet 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 paper, 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, we consider previous attempts to represent and smooth contours. In Section 3, we consider the definition of the contour representation and the mechanics of deforming the contour. In Section 4, we introduce the novel smoothing criteria and complexity scale-space. In Section 5, we consider the computation of a number of useful mathematical properties from the representation. In Section 6, we discuss the potential benefits of the paradigm for existing algorithms. In Section 7, we propose a framework for integrating the paradigm into a useful vision system. Finally, in Section 8, we present our conclusions. In the Appendix, we present the details of the computations that are necessary for completeness.

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[11], Mokhtarian & Mackworth[24] and Lowe[20]). 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[16]. 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 tend 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[20] 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[16] propose a solution to the shrinkage problem. They represent the curve by its extended circular image[15]. 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 5.4.) 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[13], Horowitz & Pavlidis[26], [17], Chen & Pavlidis[7], and Grimson[9] 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[9] pp. 105-108. The use of Θ-s space is particularly popular for object recognition systems (for example, McKee and Aggarwal[23], Perkins[27], Turney *et al*[30], Clemens[8], and Grimson[9], [10]).

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[11], the curvature estimates play an explicit role. In other cases, such as the medial axis transform[4], 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[19] 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[11], 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* [11], [12], [28] 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[1] 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[24] 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[32] 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[25]. For a mathematical treatment of curves and their properties, see Koenderink[18].

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.

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

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

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

$$\begin{aligned} x(s) &= x_c + R\cos\left(s\kappa + \theta_1\right), & (2)\\ y(s) &= y_c + R\sin\left(s\kappa + \theta_1\right), & (3) \end{aligned}$$

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

$$\left| R_1 + R_2 - \sqrt{\left(x_{c1} - x_{c2}\right)^2 + \left(y_{c1} - y_{c2}\right)^2} \right| < \epsilon. \tag{4}$$

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

$$\left| R_1 - R_2 + \sqrt{\left(x_{c1} - x_{c2}\right)^2 + \left(y_{c1} - y_{c2}\right)^2} \right| < \epsilon. \tag{5}$$

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

$$|R_1 - R_2| < \sqrt{\left(x_{c1} - x_{c2}\right)^2 + \left(y_{c1} - y_{c2}\right)^2} < R_1 + R_2. \tag{6}$$

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

$$R_1 + R_2 < \sqrt{\left(x_{c1} - x_{c2}\right)^2 + \left(y_{c1} - y_{c2}\right)^2}. \tag{7}$$

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

$$R_2 - R_1 > \sqrt{\left(x_{c1} - x_{c2}\right)^2 + \left(y_{c1} - y_{c2}\right)^2}. \tag{8}$$

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.

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. This procedure is described in greater detail in the Section A.2. Figure 1 illustrates the computation of the initial curve.

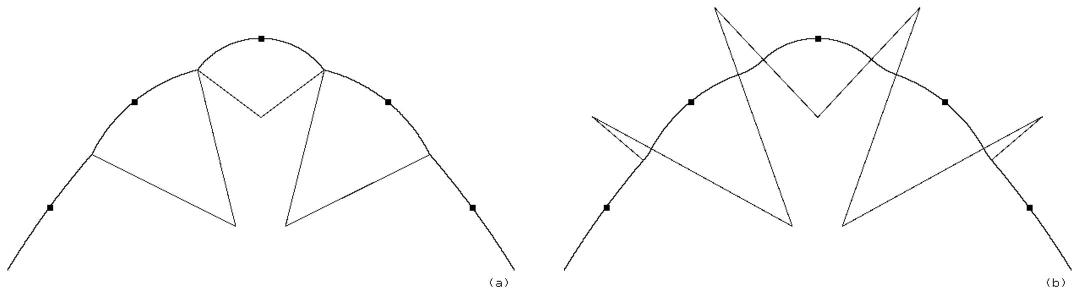

Figure 1: Initialization of a curve from the data points. The small squares indicate the location of the data points. The lines protruding from the curve represent radii of the circular arcs. Each arc in (a) passes through a data point and the midpoints between the data point and its neighbors. Radii 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.

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.

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 4, we shall consider the criteria for choosing deformations of the curve such that they lead to a more desirable curve.

3.3.1 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

$$\left|\sqrt{(x_{cn}-x_{c1})^2+(y_{cn}-y_{c1})^2}\pm R_1\right| = \left|\sqrt{(x_{cn}-x_{c2})^2+(y_{cn}-y_{c2})^2}\pm R_2\right|, \quad (9)$$

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 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[29], 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[29], 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 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 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 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 colinear. 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. A description of the procedure used to find the appropriate center point may be found in Section A.1.2. The modification of the curvature of a single arc is illustrated in Figure 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.

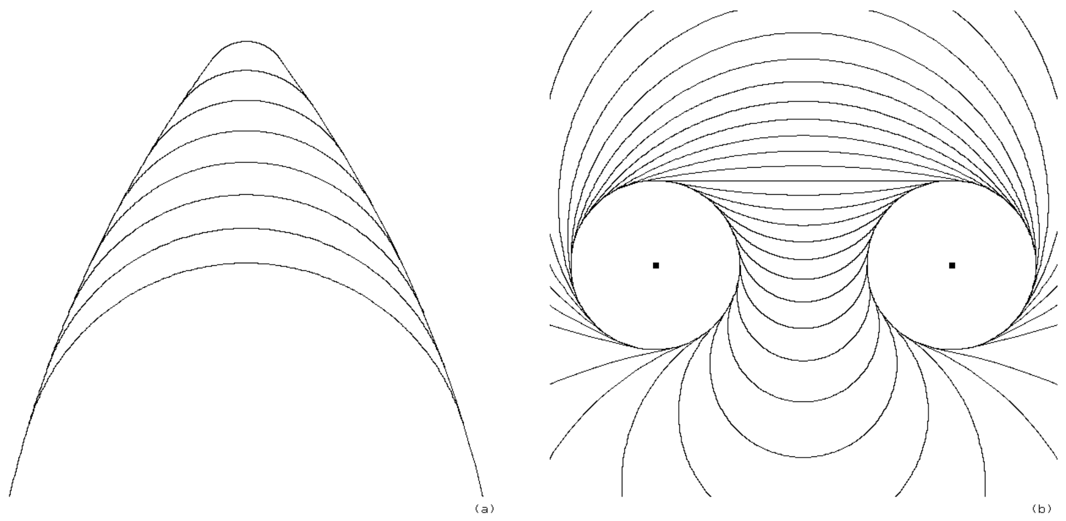

Figure 2: Deformation of the curve by changing the curvature of a single arc. In (a) the curves represent candidate arcs that are internally tangent to two neighboring arcs. In (b) the curves represent candidate arcs that are externally tangent to two circles.

3.3.2 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 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 (see Section A.1.2). 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 (see Section A.1.3). This operation is illustrated in Figure 3.

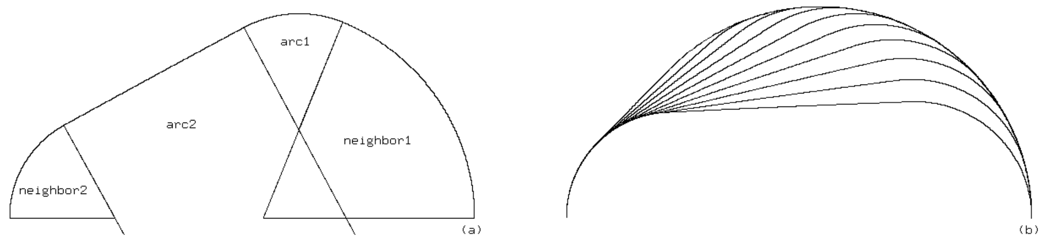

Figure 3: Deformation of the curve by rotating two neighboring arcs. An initial arrangement of two arcs and their neighbors is shown in (a). Candidate deformations of the curve involving a change in position of arc1 and arc2 are shown in (b). The curvatures associated with arc1 and arc2 are not affected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

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.

3.3.3 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 the computation. Criteria for the choice are described Section 4.3.3. 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 4.

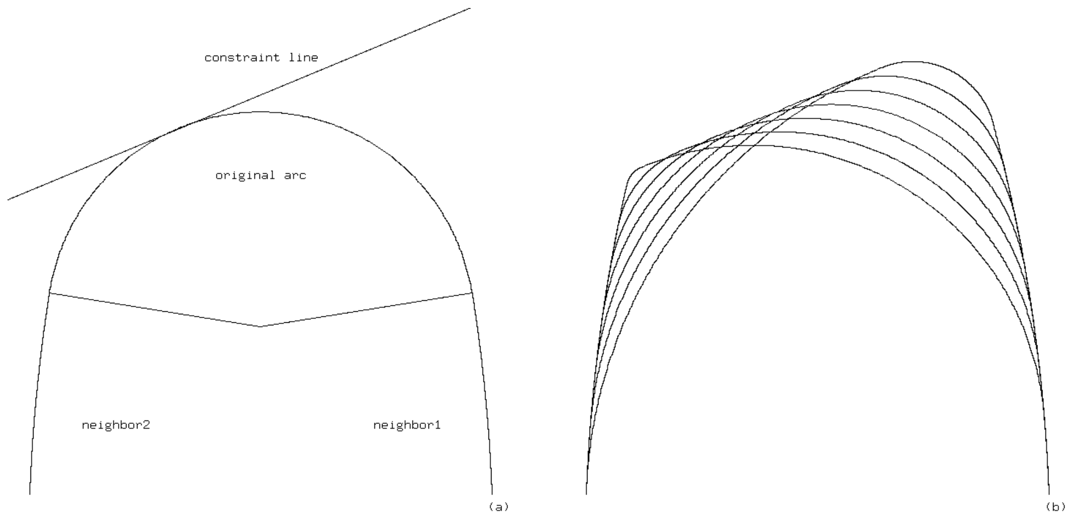

Figure 4: Deformation of the curve by splitting a single arc into two arcs. In (a) the original configuration of an arc and its neighbors is shown. In (b) each member of the family of curves consists of the two original neighbors joined by two new arcs. The point of tangency between the new arcs lies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

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.

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. Traditionally, smoothness of a contour has been measured by some function of the magnitude of the curvature[14]. However, in this paper, we consider a novel approach to this tradeoff. We propose that the complexity of the curve, rather than the magnitude of the curvature, be minimized.

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 Hoffman and Richards[11] 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.

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 3.3.1. 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 5.

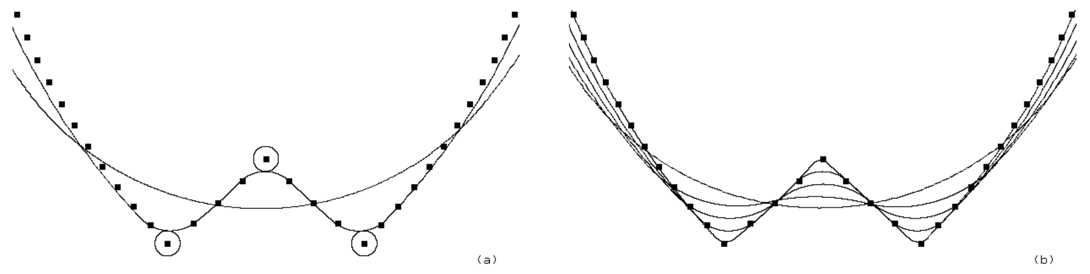

Figure 5: The reduction of the complexity of the curve. The two curves in (a) differ in their complexity measure by two. The curve that more closely follows the data points has three extrema of curvature. The extrema are indicated by the tolerance circles drawn about the critical data points. Note that 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.

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.*

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 6.

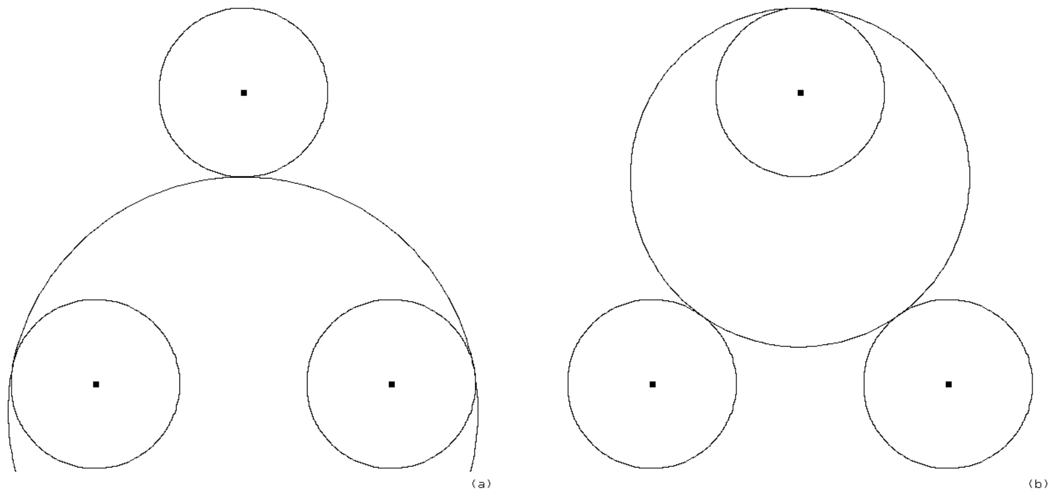

Figure 6: Determination of the upper bound of the minimum curvature and the lower bound of the maximum curvature. Three data points and their respective tolerance circles are depicted. Any curve passing within each of these tolerance circles has a maximum and minimum curvature, $\kappa_{\max}$ and $\kappa_{\min}$, respectively. The curve having the lowest value for $\kappa_{\max}$ 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 $\kappa_{\min}$ 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.

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 6a. Assume that the lower bound for the maximum curvature is given by $\kappa_{\max\ \mathrm{lb}}$. 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 $\kappa_{\min\ \mathrm{ub1}}$ and $\kappa_{\min\ \mathrm{ub2}}$, respectively. Under the condition that $\kappa_{\max-\mathrm{lb}} > \kappa_{\min-\mathrm{ub1}}$ and $\kappa_{\max-\mathrm{lb}} >$

$\kappa_{\text{min-ub2}}$, 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 7.

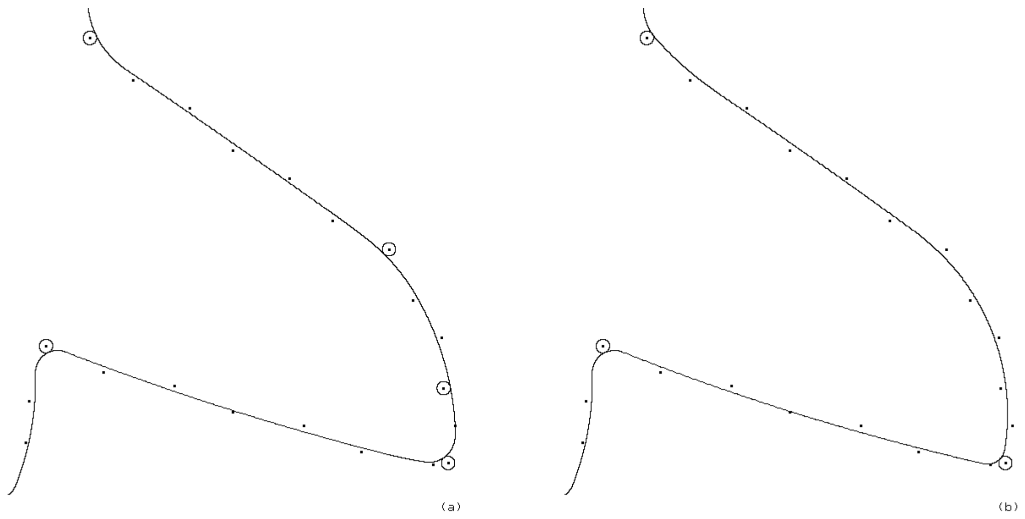

Figure 7: The elimination of an extrema pair with the aid of the verification technique. The curve in (a) is more complex than necessary given the scale parameter that is illustrated by the tolerance circles. However, the initial algorithm is unable to eliminate the unnecessary extrema pair. Upon application 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).

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 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.

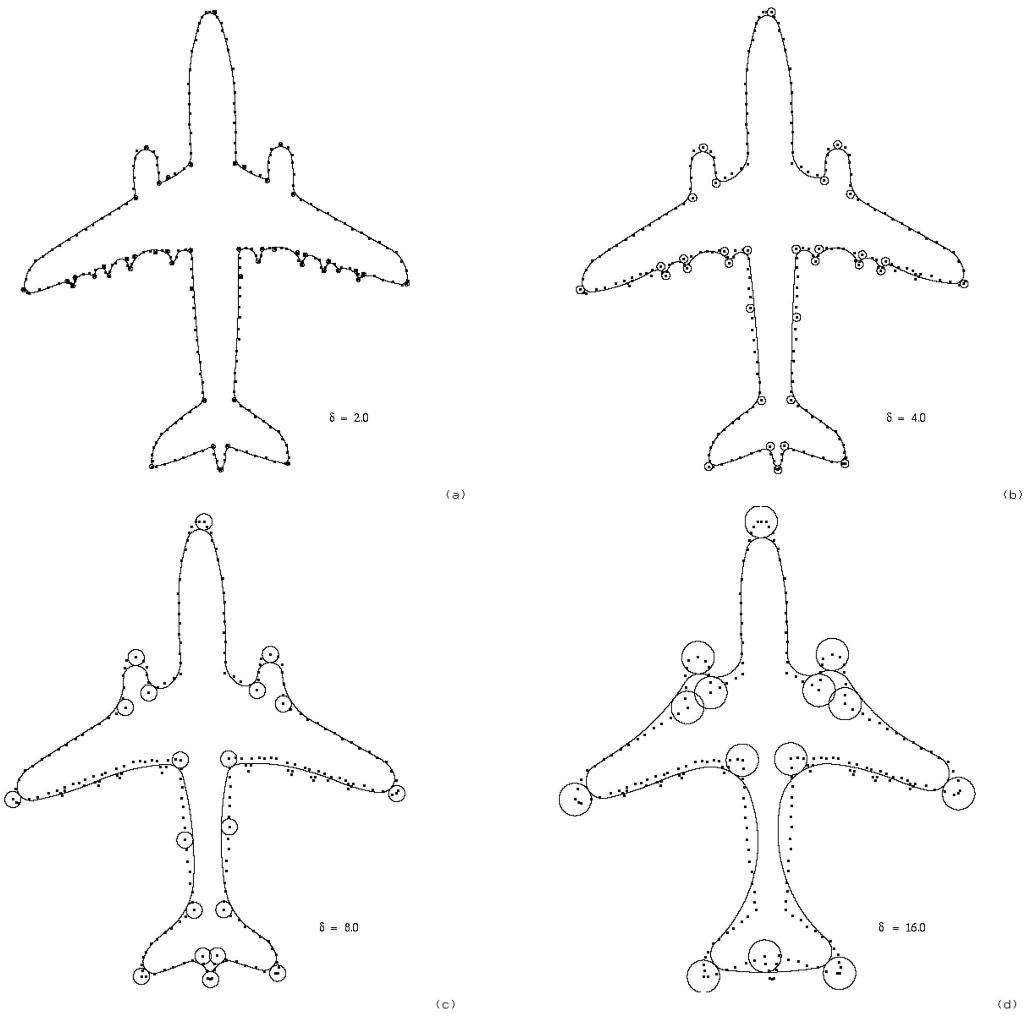

Figure 8: Minimum complexity curves for the silhouette of an airplane. Each curve is an instance of a minimum complexity curve for a particular tolerance value. The critical data points are indicated by the tolerance circle drawn about each such point. The radius of each of these circles is equal to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

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, 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 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.

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.

4.3.1 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 3.3.2, 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

$$\mathbf{T_1} = \mathbf{r_1} \times \mathbf{F_e}, \tag{10}$$

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

$$\mathbf{r_1} = \begin{vmatrix} x_{12} & - & x_{cn1} \\ y_{12} & - & y_{cn1} \end{vmatrix}, \tag{11}$$

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,

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

By combining equations 10 and 12, F_e, the scalar of the equivalent torque, may be written

$$F_e = \frac{\mathbf{T_1}}{\mathbf{r_1} \times \begin{vmatrix} \cos\theta_{12} \\ \sin\theta_{12} \end{vmatrix}}. \tag{13}$$

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

$$\mathbf{T_{t2}} = \mathbf{T_2} + \mathbf{r_2} \times \mathbf{F_e}, \tag{14}$$

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

$$\mathbf{r_2} = \begin{vmatrix} x_{12} & - & x_{cn2} \\ y_{12} & - & y_{cn2} \end{vmatrix}. \tag{15}$$

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 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 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 3.3.2, 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.

4.3.2 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

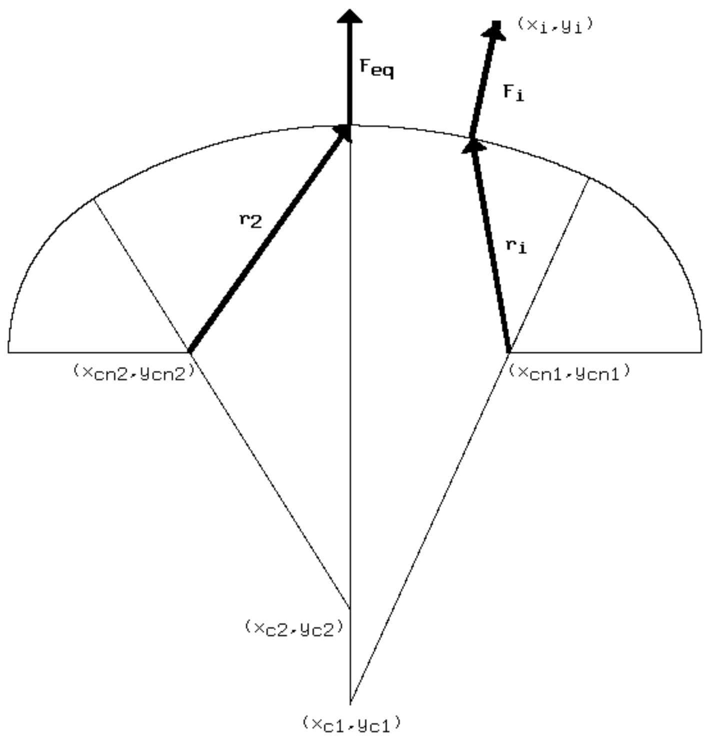

Figure 9: Torque acting to rotate neighboring arcs. Under the mechanical analogy, each data point applies a force to the curve. The resulting torque of data point (x_i, y_i) is the cross product of the force and the lever arm. Because the rotation of two neighboring arcs is coupled, it is necessary to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

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

$$\mathbf{F_i} = \left(\sqrt{(x_i - x_c)^2 + (y_i - y_c)^2} - R\right) \left| \begin{array}{c} \cos\theta_i \\ \sin\theta_i \end{array} \right|, \tag{16}$$

where θ_i is the angle of the vector from the center of the arc to the i$^{\text{th}}$ data point.

The energy related to the ith point is given by

$$E_i = \left[\sqrt{(x_i - x_c)^2 + (y_i - y_c)^2} - R\right]^2 . \tag{17}$$

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 10 illustrates this geometric constraint.

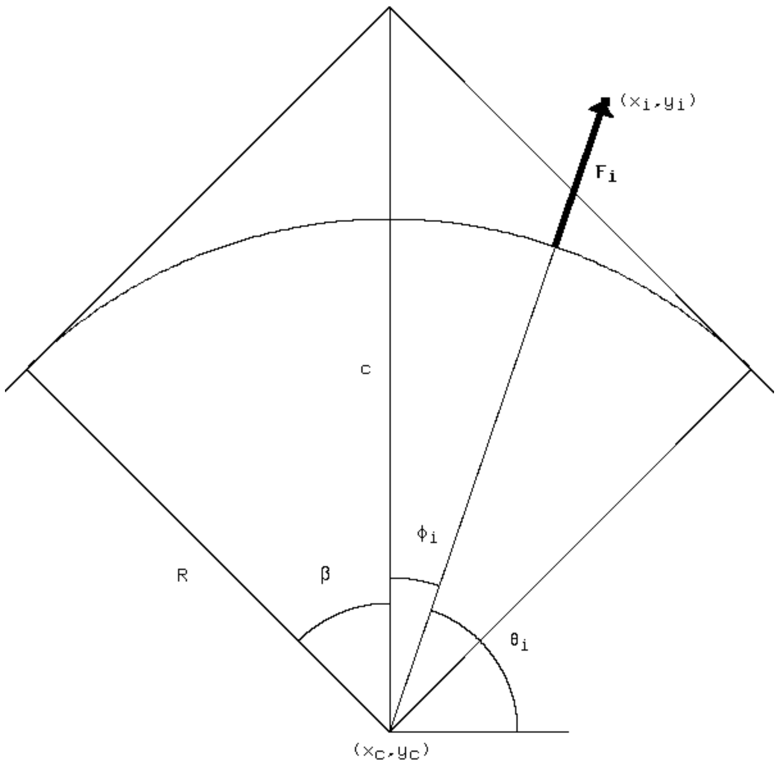

Figure 10: The force acting to deform the curvature of a single arc. The force acting on the arc by each associated data point is equal to the distance from the point to the arc. For small perturbations of the curvature, the constraint that the arc must remain tangent to its neighbors is approximated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

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,

$$F_c = \frac{dE}{ds} = 2\left(d_i - R\right)\left[\frac{dd_i}{ds} - \frac{dR}{ds}\right], \tag{18}$$

where s is the scalar distance along the constraint line. The radius of the circle 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 18 may be simplified

$$F_c = 2\left(d_i - R\right)\left(\cos\phi_i - \cos\beta\right) = 2F_i\left(\cos\phi_i - \cos\beta\right). \tag{19}$$

In the derivation of Equation 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

$$F_c = -2F_i\left(\cos\phi_i - \cos\beta\right). \tag{20}$$

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.

4.3.3 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.

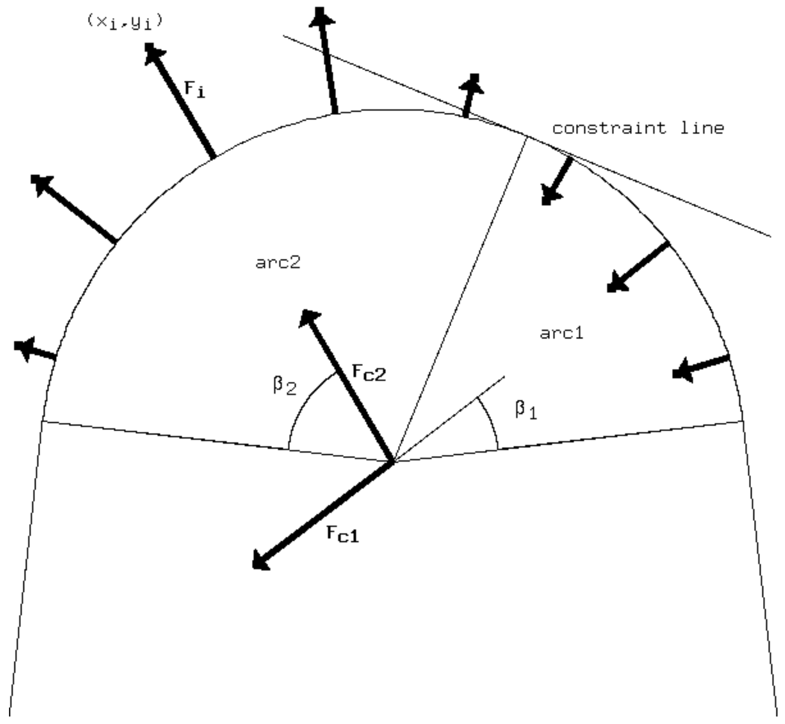

Figure 11: Setup of the operation to split an arc into two. By splitting an arc into two, it is possible to increase the curvature of one of the new arcs while decreasing the curvature of the other. For the choice of the constraint line shown, the forces are acting to decrease the curvature of arc1 and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

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 4.3.2. 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 11.

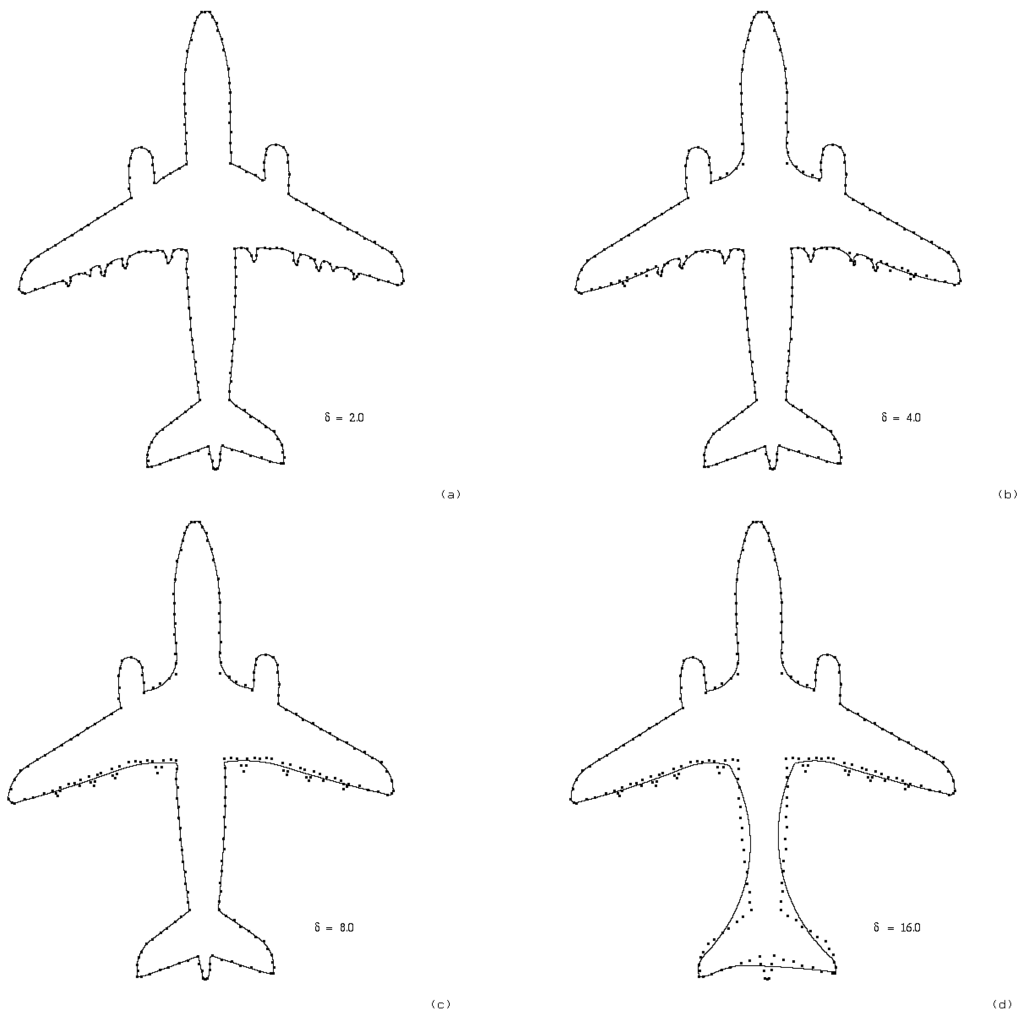

Figure 12: Minimum complexity/least-square-error curves for the silhouette of an airplane. In each case the least-square error curve that has the complexity determined from the first stage of the computation is shown. The scale parameter for each case is (a) $\delta = 2.0$, (b) $\delta = 4.0$, (c) $\delta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

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,

$$S = \frac{dE}{ds}, \tag{21}$$

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

$$S = F_{c1} \sin \beta_1 + F_{c2} \sin \beta_2, \tag{22}$$

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 22 is a result of the chain rule applied to Equation 21. As described in Section 4.3.2, 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 3.3.3. 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 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.

4.3.4 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[21]). 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 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.

5 Mathematical Properties of the Contour

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

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

$$x(s) = \begin{cases} x_{c1} & + \quad R_1 \cos\left(s\kappa_1 + \theta_{11}\right) & 0 < s < s_1, \\ x_{c2} & + \quad R_2 \cos\left(\left(s - s_1\right)\kappa_2 + \theta_{12}\right) & s_1 < s < s_2, \\ & \vdots & \\ x_{cN} & + \quad R_N \cos\left(\left(s - s_{N-1}\right)\kappa_N + \theta_{1N}\right) & s_{N-1} < s < s_N. \end{cases} \tag{23}$$

Similarly,

$$y(s) = y_{ci} + R_i \sin\left(\left(s - s_{i-1}\right)\kappa_i + \theta_{1i}\right) \qquad s_{i-1} < s < s_i. \tag{24}$$

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

$$\hat{\mathbf{n}}_{\mathbf{i}}(s) = \operatorname{sgn}(\kappa_i) \left| \begin{array}{c} \cos\left((s - s_{i-1})\,\kappa_i + \theta_{1i}\right) \\ \sin\left((s - s_{i-1})\,\kappa_i + \theta_{1i}\right) \end{array} \right| \qquad s_{i-1} < s < s_i. \tag{25}$$

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).

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

$$P = \sum_{i=1}^{N} R_i \left|\theta_i - \theta_{i-1}\right|. \tag{26}$$

The area of the interior of the curve may be computed by using Green's Theorem[29]. With appropriate application of Green's Theorem, the area, A, may be computed

$$A = \iint_R dx\,dy = \frac{1}{2}\oint_C (x\,dy - y\,dx). \tag{27}$$

This integral may be expressed

$$A = \sum_{i=1}^{N} A_i = \sum_{i=1}^{N} \frac{1}{2} \int_{s=s_{i-1}}^{s=s_i} \left(x(s) \frac{\partial x}{\partial s} \, ds - y(s) \frac{\partial y}{\partial s} \, ds \right), \tag{28}$$

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,

$$A_i = \frac{1}{2} \int_{\theta_{1i}}^{\theta_{2i}} \left(x(\theta) \frac{\partial y}{\partial \theta} \, d\theta - y(\theta) \frac{\partial x}{\partial \theta} \, d\theta \right), \tag{29}$$

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

$$\begin{aligned} A_i &= \frac{1}{2} \int_{\theta_{1i}}^{\theta_{2i}} \left((x_c + R\cos\theta)(R\cos\theta) - (y_c + R\sin\theta)(-R\sin\theta) \right) d\theta, & (30) \\ &= \frac{1}{2} \left\{ R x_c \int_{\theta_{1i}}^{\theta_{2i}} \cos\theta \, d\theta + R^2 \int_{\theta_{1i}}^{\theta_{2i}} \left(\sin^2\theta + \cos^2\theta \right) d\theta - R y_c \int_{\theta_{1i}}^{\theta_{2i}} \sin\theta \, d\theta \right\}, & (31) \\ &= \frac{1}{2} \left\{ R x_c \left[\sin\theta \right]_{\theta_{1i}}^{\theta_{2i}} + R y_c \left[\cos\theta \right]_{\theta_{1i}}^{\theta_{2i}} + R^2 \left[\theta_{2i} - \theta_{1i} \right] \right\}. & (32) \end{aligned}$$

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

5.4 Extended Circular Image

The extended circular image has been proposed as a useful description of the shape of a curve[16]. The value of a particular point of the extended circular image, $C(\psi)$, for a convex curve is

$$C(\psi_a) = \frac{1}{\kappa(s_a)}, \tag{33}$$

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.

6 Discussion

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 these properties; the estimates are typically extremely noisy. However, the representation presented here yields 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[11] 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[1]. 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[24]. In their approach, the shape of the contour is characterized by the positions of 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 this 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.

The contour representation also improves the ability to compute the medial axis transform[4] 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[3], for example). Given the ability to reduce the effects of noise on the contour, it is possible to compute the skeleton more reliably.

The complexity scale-space introduced here also has direct implications for the medial axis skeleton. A reduction in the number of extrema of curvature of the contour typically leads to a reduction in the number of branches of the skeleton. The number of branches of the skeleton is a reasonable measure of the complexity of the skeleton. Thus, it is natural to develop a complexity scale-space for the medial axis transform. We consider this concept further in a companion memo[6].

The minimum complexity smoothing paradigm has a variety of advantages over existing 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.

Another advantage of the minimum complexity paradigm is that it provides a *true* scale-space representation of the contour. The most desirable property of a scale-space representation is that finer scales possess a greater amount of detail than is present at the more coarse scales. The complexity measure explicitly guarantees that this is the case. The number of features present in the contour decreases as the scale becomes more coarse.

Most other approaches, such as those that rely on Gaussian filters with different spatial widths, are more accurately described as resolution-spaces. As the standard deviation of the Gaussian filter increases, the resolution diminishes. That is, the ability to localize the features diminishes. However, the level of detail, as measured by the number of salient features, may or may not decrease.

The advantage of a true scale-space, in this sense, is that whenever a feature is present, it is localized as accurately as possible. Furthermore, when a feature is not present, it is completely eliminated from in the representation. There is no advantage in representing features with less accuracy than possible, as is done in the more coarse resolutions of a resolution-space.

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.

7 Future Work

The use of the analytical representation of contours holds promise for improving the capabilities of machine vision systems. We have discussed the improvement to existing algorithms. Unfortunately, as is often the case, the obvious, naive extensions of these algorithms contain numerous pitfalls. In this section, we consider the most glaring problem. Then, we propose a framework for integrating the representation into a useful vision system.

It is wishful thinking to believe that the outlines of objects will be provided to a recognition system. Edge detection, as it has been traditionally defined, does not provide a robust description of the physical discontinuities present in the image. Furthermore, there is presently no reliable way of connecting edges together to form coherent, meaningful contours. The Canny[5] edge detector, for example, yields only sparse, unconnected data points. Following the data points may yield fragments of a meaningful contour, but this almost never yields a complete outline of an object.

However, the paradigm described in this memo does provide capabilities that are useful for incorporation in vision systems. Although the paradigm does not interface well with current early vision processes, this does not preclude development of more advanced feature extraction algorithms. It is possible, if not likely, that more robust methods for obtaining salient features will emerge to replace traditional edge detection. While these features are unlikely to demark complete objects explicitly, it is likely that they will be less primitive than collections of edge points. If the features consist of contours or regions (each implicitly bounded by a contour), the paradigm described here will be helpful for representing and interpreting the shape of the features. Improvement of the representation of these hypothetical features would necessarily lead to improvement in the ability to synthesize reasonable interpretations of images, recognize objects, and track the features for the purpose of motion estimation.

In future work, I plan to develop an alternate feature set based upon salient regions bounded by zero-crossings of the Laplacian of Gaussian filter. While zero-crossings have been proposed as edge locations in the past (most notably, Marr and Hildreth [22]), they do not provide robust results. In areas of the image where contrast is low, the zero-crossings tend to wander off the subjective location of the physical discontinuity. Rather than enclosing regions of the image that are subjectively of similar brightness, such regions tend to blend or merge together. Figure 13 illustrates this phenomenon.

Typically, the zero-crossing contour has an extremum of curvature at or near the location where neighboring salient regions appear to blend together. Thus, by breaking the contour at the extrema of curvature, as suggested by Hoffman and Richards[11], it may be possible to decompose larger regions bounded by zero-crossings into smaller, simpler regions. These simpler regions are more likely to

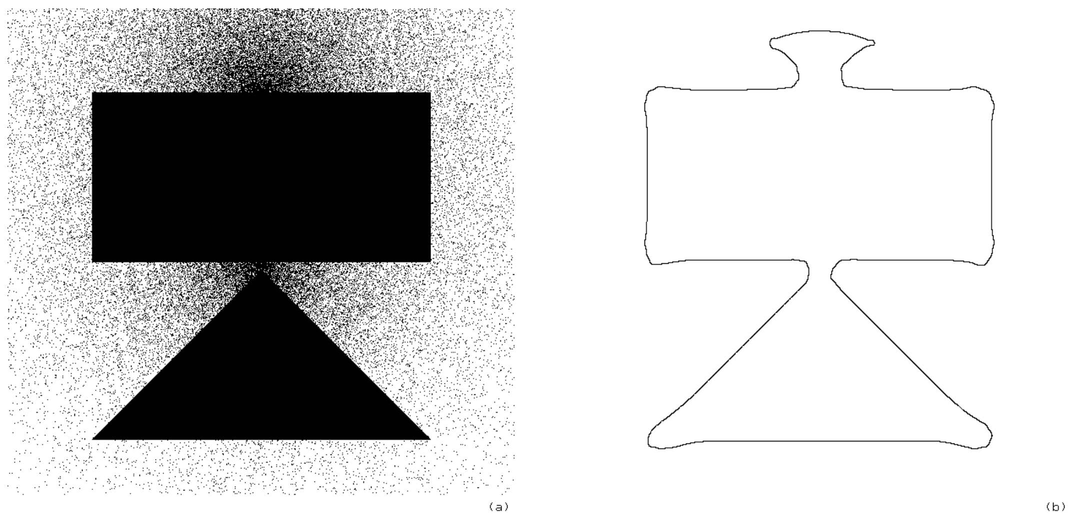

Figure 13: Zero-crossings of the Laplacian of Gaussian filter. Ideally, the zero-crossings of the Laplacian of Gaussian filter demark discontinuities in the brightness of the image. However, it is well known that in areas of low contrast, the zero-crossings often wander away from the subjective location of the discontinuity. Furthermore, nearby regions tend to interfere with one another such that the subjective regions blend together. A synthetic image of a rectangle and a triangle on a textured background is shown in (a). The zero-crossings of a Laplacian of Gaussian filter illustrate the undesired effects in (b). Notice that the zero-crossings wander away from the rectangle at the top and the triangle blends together with the rectangle.

correspond to subjective, salient features in the image. We call such regions *simple region features*.

Simple region features consist of the region bounded by the zero-crossing as well as the zero-crossing itself. Thus a measure of the strength or significance of the feature may be obtained from the average power of the signal contained within the region. Conversely, the significance of edge points is typically measured by comparing the values of a pair of points adjacent to the zero-crossing. Because of this greater spatial support of the significance measure, the acquisition of simple region features is likely to be more robust and reliable than the acquisition of edge points.

As we describe in a companion memo[6], the analytical representation lends itself quite well to the computation of the medial axis skeleton. Such a capability is also beneficial for the computation of simple region features. The skeleton could be used as an alternate means of breaking the complex regions into simple region features. Each branch of the medial axis skeleton corresponds to a salient portion of the region. Hence, each of these subregions may act as a simple region feature.

We consider the future development of simple region features here because the contour reprepresentation described here is the key ingredient necessary to make their computation possible. The analytic representation paradigm allows for mathematical treatment of the contour and in particular allows for the computation of reasonable estimates of curvature. In addition, the complexity scale-space provides the basis for a multiple complexity decomposition of the larger regions into simple region features. Furthermore, the discussion provides a retrospective motivation for the development of the contour representation paradigm.

8 Conclusion

In this paper, 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 the medial axis transform. 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. The improved representation and smoothing capability for contours will lead to better interpretation and recognition results. Furthermore, the paradigm is promising for the development of a more robust feature set than is currently available.

References

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

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

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

[4] 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.

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

[6] Ronald D. Chaney. Complexity as a scale-space for the medial axis transform. AI Memo 1397, MIT Artificial Intelligence Laboratory, November 1992.

[7] 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.

[8] D. T. Clemens. The recognition of two-dimensional modeled objects in images. Master's thesis, Massachusetts Institute of Technology, Dept. of Elec. Eng. & Comp. Sci., 1986.

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

[10] 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.

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

[12] 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.

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

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

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

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

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

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

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

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

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

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

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

[24] 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*, 8(1):34–43, Jan. 1986.

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

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

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

[28] 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.

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

[30] 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.

[31] Paul Wallach. Garbage in, garbage out: Simple geometry brings supercomputers to their knees. *Scientific American*, page 126, Dec. 1990.

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

Appendix

A Computational Details

In this appendix we consider the details of the computation of the curve representation.

A.1 Geometric calculations

In this section we consider several simple geometric calculations involving circles. First, we consider the computation of the center point of a circle tangent to another known circle when the direction of the point of tangency from the known circle and the radius of the unknown circle are specified. Next we consider the computation of a circle that is tangent to two known circles when the direction from one center point to the unknown center point is specified. Finally, we consider the computation of a circle tangent to two known circles when the radius of the unknown circle is specified.

A.1.1 A circle tangent to another circle

Determining the center point and radius of a circle (circle2) under the constraint that it be tangent to a fixed circle (circle1) is a common required for the contour representation paradigm. We assume that the direction of the tangent point from the center of the fixed circle is given in the form of an angle, β. Furthermore, a desired value of the radius (and curvature) of the new circle is also specified. The difference between the desired value of the radius, R_{2d} and the value ultimately computed, R_2 is due to roundoff error.

First, It is necessary to compute the distance, d_{12} between the between the centers of circle1 and the circle2. If the circles are to be externally tangent, the distance is given by

$$d_{12} = R_1 + R_{2d}. \tag{34}$$

If the circles are to be internally tangent,

$$d_{12} = \left| R_1 - R_{2d} \right|. \tag{35}$$

These relations follow from the definitions of internal and external tangency described in Section 3.1

The position of the center of circle2 may be computed from the distance between the circles and the direction of tangent point from circle1. The center point of circle2 (x_{c2}, y_{c2}) is given by

$$\begin{aligned} x_{c2} &= x_{c1} \pm d_{12} \cos\beta, & (36) \\ y_{c2} &= y_{c1} \pm d_{12} \sin\beta. & (37) \end{aligned}$$

The minus signs are appropriate when the circles are internally tangent and $R_{2d} > R_1$. Otherwise, the plus signs are appropriate.

Finally the value of the radius of circle2 is computed. If the circles are internally tangent and $R_{2d} > R_1$, the radius is given by

$$R_2 = R_1 + \sqrt{(x_{c2} - x_{c1})^2 + (y_{c2} - y_{c1})^2} + R_1. \tag{38}$$

Otherwise, the radius is given by

$$R_2 = \left| R_1 - \sqrt{(x_{c2} - x_{c1})^2 + (y_{c2} - y_{c1})^2} \right|. \tag{39}$$

Mathematically, $R_2 = R_{2d}$. However, it is important to use the value of R_2 computed above. Otherwise, roundoff error might lead to a result in which circle1 and circle2 are not tangent.

A.1.2 A circle tangent to two circles constrained by direction

Another useful operation is the computation of a circle that is tangent to two other known circles. We refer to the unknown circle as circle2 and the known circles as circle1 and circle3, respectively. The direction from the center point of circle1 to the tangent point of circle2 is specified by angle β.

If two circles are tangent, their respective center points and the point of tangency are colinear. Therefore, the center point of circle2 is constrained to lie along the line specified by

$$\begin{aligned} x_{c2} &= x_{c1} + s\cos\beta, & (40) \\ y_{c2} &= y_{c1} + s\sin\beta, & (41) \end{aligned}$$

where s is a parameter of the line. Conceptually, the objective is to find the value of s that corresponds to the center point of a circle that is mutually tangent to circle1 and circle3 with the appropriate type of tangency.

The specific calculation required to find the appropriate circle is dependent upon the relationship between circle1 and circle3 as well as the sign of curvature of the arcs associated with circle1 and circle3. Because there are several such cases, we consider one case and generalize. Assume that the arcs associated with circle1 and circle3 have the same sign. This implies that circle2 has the same tangency type with circle1 that it has with circle3. Furthermore, assume that circle1 and circle2 are external.

Under these assumptions, there are two possibilities that must be considered. The first possibility is that circle2 is internally tangent to circle1 and to circle3. The other possibility is that circle2 is externally tangent to circle1 and circle3.

First, we consider the possibility that circle2 is internally tangent to both circle1 and circle3. The algorithm iterates over candidate values of the radius of circle2.

For each such value, a candidate circle2 is computed that is internally tangent to circle1 as described in Section A.1.1. Based upon the relationship of the candidate circle with circle3, it is possible to determine whether the candidate radius is too large or too small. The next candidate value of the radius is chosen appropriately.

For each candidate circle, it is possible to determine whether the radius is larger or smaller than that of the desired circle2. If the candidate circle is external to circle3, intersects circle3, or is externally tangent to circle3, the radius is too small. If circle3 is internal to the candidate circle, the radius is too large. If the desired circle2 is actually externally tangent to circle1 and circle3, this test always leads to the conclusion that the candidate radius is too small.

To begin the iteration, the algorithm considers successively larger values for the radius until an upper bound, $R_{\max}$, and a lower bound, $R_{\min}$, is found. The first candidate radius, R, is chosen arbitrarily to be twice the radius of circle1. The candidate circle is computed as described above. It is possible to determine whether the candidate radius is too large or too small depending on the relationship of the candidate circle to circle3. If the candidate radius is too large, its value is assigned as $R_{\max} = R$ and the algorithm moves to the next step. If the value of R is too small, its value is assigned to the lower bound $R_{\min} = R$ and a new value for the candidate radius that is twice the previous value is attempted. This procedure is repeated until a value of the upper bound if found. If the candidate radius becomes larger than a specified maximum, it may be concluded that circle2 is actually externally tangent to circle1 and circle2.

Once the upper and lower bounds for the radius have been determined, the algorithm iterates to narrow the gap between the two bounds. On each iteration, the candidate radius is chosen to be the average of the two bounds. If the candidate radius if found to be too small, the lower bound is updated appropriately. Conversely, if the radius is found to be too large, the upper bound is updated. Thus, on each iteration the difference between $R_{\min}$ and $R_{\max}$ is cut in half. This procedure is repeated until a candidate value of the radius yields a circle that is internally tangent to circle3.

If it is determined that circle2 is externally (rather than internally) tangent to both circle1 and circle3, a similar procedure is used to determine the appropriate radius. The initial upper and lower bounds are determined by doubling the candidate radius until the upper bound is found. The algorithm narrows the gap between the upper and lower bound until the appropriate circle is found. The only significant difference in the computation is the test to determine whether the candidate radius is too large or too small. In this case the candidate radius is too small if the candidate circle is external to circle3. If the candidate circle intersects circle3, circle3 is internal to the candidate, or the circles are internally tangent, the radius is too large.

The difference between the specific cases that we have considered here and others is primarily the test to determine whether a candidate radius value is too large or too small. In each case, the test is similar in form to that given above - test is based upon the relationship between the candidate circle and circle3. However, the specific relationships between the circles that lead to a particular conclusion differs.

A.1.3 A circle tangent to two circles constrained by curvature

Consider the computation of a circle with specified curvature that is tangent to known circles. Again, we refer to the unknown circle as circle2 and the known circles as circle1 and circle3, respectively. The curvature and, therefore, the radius of circle2 are specified. It is desirable to find the center point of circle2 such that the circle is mutually tangent to circle1 and circle3.

Again, the sign of curvature of each circle places constraint on the possible solution. That is, the signs of curvature of two tangent arcs must be consistent with the tangency type. Given these constraints, there are typically two solutions. The two valid center points for circle2 are symmetric about the line determined by the center points of circle1 and circle2. However, it is almost always possible to choose the more desirable solution based upon the context of the operation.

Now, we consider a particular case for this operation. Later, we shall generalize. In particular, consider the case where circle1 and circle3 are external and circle2 has a sign of curvature that is opposite to both signs of curvature of circle1 and circle3. Thus, circle2 is externally tangent to both circle1 and circle3.

The direction of the center of circle3 from the center of circle1 is given by the angle $\beta_{\min}$ and

$$\beta_{\min} = \arctan\left(y_{c3} - y_{c1},\ x_{c3} - x_{c1}\right). \tag{42}$$

The opposite direction may be specified as β_{rmmax} and

$$\beta_{\max} = \arctan\left(y_{c1} - y_{c3},\ x_{c1} - x_{c3}\right) = \beta_{\min} + \pi. \tag{43}$$

The names $\beta_{\min}$ and $\beta_{\max}$ are chosen because these angles represent convenient endangles for the iteration to find the position of circle2. That is, the direction of the tangent point between circle1 and circle2 is between $\beta_{\min}$ and $\beta_{\max}$. Note that, for simplicity, we have tacitly chosen between the two symmetric solutions. The other solution lies between $\beta_{\max}$ and $\beta_{\min}$ (modulo 2π).

The algorithm iterates over candidate values of the angle of tangency between circle1 and circle2. On each iteration β is computed such that it bisects $\beta_{\min}$ and $\beta_{\max}$. A candidate circle is computed consistent with this tangent direction as described in Section A.1.1. Depending on the relationship between the candidate circle and circle3, either $\beta_{\min}$ or $\beta_{\max}$ is given the value of β. The algorithm iterates until a candidate circle is found that is externally tangent to circle3.

For the case described, the value of $\beta_{\min}$ is updated with the value of β whenever the candidate circle intersects circle3. The value of $\beta_{\max}$ is updated with the value of β whenever the candidate circle is external to circle3. The test for narrowing the interval for β is similar for other cases. The difference lies in the particular relationships between the candidate circle and circle3 that results in a change of $\beta_{\min}$ versus $\beta_{\max}$.

A.2 Curve initialization

The curve is initialized in two steps. First, for each data point a circle is computed that passes through the point. Second, for each pair of neighboring arcs, a circle is computed that is tangent to each neighbor. The points of tangency delimit the range of each arc of the circle. The result is a chain of pairwise tangent circular arcs that pass through each data point. The initial curve is somewhat arbitrary, but it is to be immediately modified by the smoothing algorithm.

Consider the first step: the computation of circle that passes through a particular data point. It is desirable that the circle be chosen is such a way that it may be joined easily to its neighbor with an arc computed in the second step. A reasonable choice is the circle that passes through the data point and each midpoint between the data point and its respective neighbor.

Each line segment determined by the data point and one of the neighbor midpoints is a chord of the initial circle. The perpendicular bisector of a chord passes through the center of a circle. Therefore, the center of the circle is determined by finding the intersection of the perpendicular bisectors of the two chords. The radius is the distance from the center to the data point.

The sign of curvature of the initial arc is determined from the data point and its two neighbors. Consider the line that is determined by the two neighbor points. If the point of interest is on the right side of the line as an observer moves from the preceding neighbor point to the following neighbor point, the curvature of the initial arc is positive. Conversely, if the point is to the left of the line the curvature of the initial arc is negative. Mathematically, this relation may be expressed as

$$\operatorname{sign}(\kappa) = \operatorname{sign}\left((x_p - x_{n1})(y_{n2} - y_{n1}) - (y_p - y_{n1})(x_{n2} - x_{n1})\right), \qquad (44)$$

where κ is the curvature of the initial arc, (x_p, y_p) is the center point and (x_{n1}, y_{n1}) and (x_{n2}, y_{n2}) are the neighboring points.

Consider the second step. For each neighboring pair of circles that pass through their respective data points, a circle must be computed that is tangent to each neighbor. The radius of this intermediate circle is arbitrarily chosen such that it is half the distance between the neighboring pair of data points. This choice of radius guarantees that the data point will be within the sector of its associated arc. Because the intermediate circle must be tangent to each neighboring circle, there is only one legal position of the center point. The position of the center of the circle is determined using the procedure described in Section A.1.3.

After the second step has been performed for all data point pairs, the result is a list of pairwise tangent circular arcs. The endangles are determined by the points of tangency between the neighboring arcs. The result is a legal curve, as defined above, that passes through each data point in order.

A.3 Straight Lines

All previous discussion has assumed that the radii of all circular arcs are finite. However, in the limit where the radius of a particular arc tends to infinity, the arc approaches a line segment. The curvature, in this case, tends to zero.

Because it is not possible to represent the arc as having an infinite radius with a center point infinitely far away, the algorithm treats zero curvature arcs as a special case. Such a "straight arc" is defined by its endpoints, the points tangent with its neighboring arcs.

It is necessary to define geometric properties for the straight arc. The distance from any point to a line is the distance between the point and the projection of the point onto the line. Ideally, a line is tangent to a circle iff the distance between the center of the circle and the line is equal to the radius of the circle. To account for quantization, a line is tangent to a circle iff

$$|d - R| < \epsilon, \tag{45}$$

where d is the distance from the center point to the line and R is the radius of the circle. A line intersects a circle iff the line and circular are not tangent and the distance between the center point and the line is less than the radius. A line is external to the circle iff the line and circle are not tangent and the distance between the center point and the line is greater than the radius.

Deformation of the curve in the case of straight arcs is a straightforward extension of the operations described above. The mechanical analogy applies equally well to straight arcs as to circular arcs. However, when the position of the neighbor of a circular arc is to be modified, it is appropriate to compute the component of the force in the direction of the straight arc, rather than the torque (the motion of the center of the arc is a translation along a line parallel to the straight arc).