Planar Curve Representation and Matching

Maher Al-Khaiyat and Farhad Kamangar
Computer Science and Engineering Department
University of Texas at Arlington
Arlington, Texas 76019, USA
[al-khaiy|kamangar]@cse.uta.edu

Abstract

In this paper, we discuss a method for representing and matching planar curves. The technique is based on using calculations from concentric circles to represent each curve by two sets of angles. The angles are defined by vectors constructed from the center point of the circles and the points on the curve trace that intersect each circle. The circles have incrementally increasing radii represented by the minimum radius and the radius increment value. The number of circles used specifies the level of abstraction at which the curves are represented. This representation is invariant to translation and rotation transformations. Experiments with different classes of curves have shown that our technique is robust to digitization errors and noise effects, and can perform well when the number of concentric circles are relatively small. In particular, we describe the potential applicability of this technique to fingerprint identification problem.

1 Introduction

Curve matching is the approach used to find the best fit between two curves. By best fit, we mean that two curves can be aligned together such that a section or subcurve from one curve is geometrically similar to that of the other curve. The importance of curve matching is due to its wide applicability in computer vision and object recognition systems. Curves represent boundaries of objects, lines in fingerprints, and contours in maps. Relationships between different objects can be mapped into relationships between the curves extracted from these objects.

This paper presents an approach to representation and matching of two-dimensional planar curves using concentric circles information. The approach is invariant to translation and rotation transformations and does not require the curves to possess critical points or breakpoints. In addition, the computational complexity of matching phase is $O(M)$ where M is the number of circles used for the representation and is much less than n (number of points on the curve). It has a hierarchical nature in which curves can be represented using hierarchy of abstracts. This property promotes the parallelization of matching to improve performance and eliminate unnecessary testing.

Previous research on curve representation and matching ([1]-[11]) can be categorized into several classes. Wolfson [1] presented two algorithms for curve matching that fall under the curvature-related [12] class of algorithms. His algorithms used the characteristic strings to represent the curves. The characteristic strings of two curves are matched together to find a set of possible candidate matches. The candidate matches are mapped back to the original curves where error is calculated for each pair. The best result is returned. The methods are translation and rotation invariant. The two algorithms differ in matching of characteristic strings. While the first one uses a simple string-matching algorithm that requires one more step to quantize the characteristic strings, the second method maintains shift accumulators and considers a tolerance measure ε. The latter is more robust but has an $O(n^2)$ worst-case complexity. Our matching algorithm uses the more compact concentric circles representation for matching and its complexity depends linearly on the length of the representation vector which is less than that of the characteristic strings.

Another approach suggested by Freeman [5] applies to closed or open curves. We only consider the open curve problem in our approach, but suggest scene-matching approach that can apply to closed curves. Freeman's approach belongs to the feature-calculation class of algorithms. For open curves, his approach uses discontinuities in curvature (DICs) [13] to break curves into segments and establish correspondence between different segments from different curves. For each segment, the features calculated are:

1. Length of segment divided by the distance between end points of segment.
2. Total "bay" area divided by square the distance between end points of segment.
3. Total "peninsula" area divided by square the distance between end points of segment.
4. Maximum "bay" depth divided by the distance between end points of segment.
5. Maximum "peninsula" depth divided by the distance between end points of segment.

The above features are translation, rotation and scale invariant. The invariance to scaling results from normalizing by the distance between end points for the length features and square that distance for the area features. Our concentric circles representation finds breakpoints as a derivative of the algorithm and does not require their existence beforehand.

The third class, template-based approach, was introduced by Turney et al. [14] for recognizing objects using their boundary curves. Objects exist in a scene and may be partially occluded. Every object is represented by a set of templates. A template is a two-dimensional boundary curve of the object and is represented by a set of "salient features". Each salient feature (also known as subtemplate) is distinct on the template and distinguishable from other templates. It has a weight factor (called "saliency measure") that measures its distinctiveness. If τ_i is a subtemplate of template T, and w_i is the weight associated with it, then

$$\sum_{i=1}^{|\tau|} w_i = 1 \quad , w_i \in [0,1] \tag{1}$$

where $|\tau|$ is the number of subtemplates of T.

Finding the saliency is accomplished by correlating every template against all other possible templates. A database is maintained to keep all templates and subtemplates in the system. These features are not local to the object itself. Thus, the weights need to be updated whenever a new object is considered and only those in the system can participate in the matching and recognition phases. The method is useful in product assembly lines where objects available to the system are known.

In the following sections, we present the concentric circles representation and matching algorithms. We will also present fingerprint recognition work as a potential application of the algorithm and conclude with ongoing research and future work.

2 Concentric Circles

2.1 Preprocessing

In the preprocessing phase, captured gray-scale images were smoothed using an "edge-preserving" smoothing technique. Four spatial averaging filters in the neighborhood of $w \times w$ were used. These filters calculate the averages along the horizontal line, vertical line, and the diagonals passing through the pixel whose value is to be updated. The algorithm updates the pixel value with the average that results in minimum variance. This technique is sometimes called "averaging by minimum variance". Binarization is achieved by a threshold value. We adopted a local thresholding approach. Thinning or skeletonization of curves is used to obtain 1-pixel wide curves. The thinning algorithm preserves the 8-neighborhood connectedness while eliminating redundant pixels.

2.2 Representation

Concentric circles representation uses 8-neighborhooh chain codes that encode trace of curves as input. From that, a curve is represented by two sets of angles $\Phi = \{\phi_i : 1 \le i \le M\}$ and $\Psi = \{\psi_i : 1 \le i \le M\}$ corresponding to angles between vectors centered at a point, p_0, on the curve trace. The procedure is to draw circles centered at p_0 with incrementally increasing radii. The points at which these circles intersect the curve are marked and used as the tip points of vectors initiated from p_0. Each vector will have a magnitude equal to the radius of the circle at which the intersection occurred. It is important that the curve intersects any circle at two points only. When a situation occurs that a curve intersects a circle at more than one location on each side of the center point, the curve will be segmented. In fact, this is the criterion we will use to segment the curves, and each segment will be represented by the two sets of angles between the vectors at every circle. The circles will have a minimum radius, a maximum radius, and an increment by which two consecutive circles differ in radius.

The angles for a given circle c_i whose radius is r_i are calculated as follows: find two points on the curve segment intersecting this circle. Mark these points p_i^+ and p_i^- depending on their location with respect to p_0. The angle between vectors $\overline{p_0 p_i^+}$ and $\overline{p_0 p_i^-}$ is the i th element ϕ_i (indexed by r_i) in the first set. For the i th element of the second set, the angle ψ_i between $\overline{p_0 p_{i-1}^+}$ and $\overline{p_0 p_i^+}$ is chosen. $\overline{p_0 p_{i-1}^+}$ is the vector from the circle whose radius is r_{i-1} ($r_i - r_{i-1} = \Delta r$ and is considered constant, but it is not required to be so). The angles are graphically represented in Figure 1.

Given a segment of some curve, the algorithm consists of two steps. The first step is to choose the center point of all the circles. The second step is to calculate the representation of the curve segment. The following are the two steps, and the procedures involved in both of them.

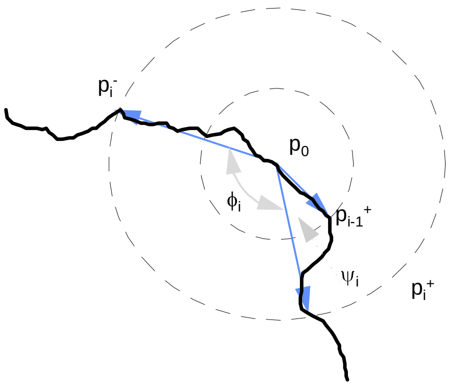


Figure 1: Graphical demonstration of the concentric circles representation angles.

- Determine p_0 on the curve segment.

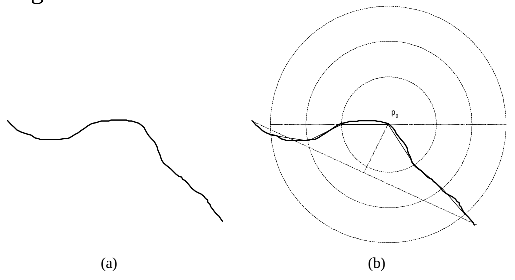
1. Connect the two end points of the curve segment.
2. At the mid point of the connecting line, construct a perpendicular line. p_0 is the point on the curve segment intersecting that perpendicular.

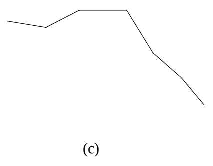
- Determine the curve representation.

1. Initialize $r_i = r_{\min}$.
2. Repeat the following until $r_i = r_{\max}$ or no more points are found
   a. find the furthest point p_i^+ to the right of p_0 such that the length of $\overline{p_0 p_i^+}$ segment is less than or equals to r_i . If it is less than r_{i-1}, then stop.
   b. find the furthest point p_i^- to the left of p_0 such that the length of $\overline{p_0 p_i^-}$ segment is less than or equal to r_i . If it is less than r_{i-1} , then stop.
   c. calculate ϕ_i as the angle between $\overline{p_0 p_i^+}$ and $\overline{p_0 p_i^-}$.
   d. calculate ψ_i as the angle between $\overline{p_0 p_{i-1}^+}$ and $\overline{p_0 p_i^+}$.
   e. increment r_i by Δr .

Figure 2 demonstrates three circles that were used for the representation of a curve. In this example, we overlaid the circles on the original curve at point p_0 . Using the same figure, one can identify the vectors and calculate the radius-angle information based on the above algorithm. We calculated the angles and tabulated the representation in Figure 2(d). According to this information, we were able to recreate the points on the curve that intersected the circles.

To reconstruct the curve, we used straight-line segments to connect every two consecutive points (Figure 2(c)). No information about the curve segment between these points can be inferred from the representation. A large increment value of radius means that more information about the curve segment is abstracted out or lost. This radius incremental value is important to preserve features of the curve and minimize error. When it is small, features will be detected and we will be able to reconstruct them, but redundant information such as noise will be included. When it is large, some features are lost or abstracted out but noise is suppressed. Figure 3 shows a hierarchical representation of a curve to demonstrate the effect of radius increment value. The curve is first represented using one circle. The radius is chosen so that the curve is enclosed by





r_i	ϕ_i	ψ_i
1	57°	0°
1.75	118°	7°
2.5	124°	-1°

(d)

Figure 2: Concentric circles representation. (a) Original curve. (b) Representation circles. (c) Reconstructed curve. (d) Representation sets.

the circle. Then, the radius is divided by two and the curve is represented using two circles. After that, each radius is again divided by two and the curve is represented using four circles, and so on. Each case is equivalent to representing the curve at one level of abstraction. The first case, representation using one circle, is the most abstract representation. We used four levels of abstraction to represent the curve, and reconstructed it at each level. Note that the level number is inversely related to the radius increment value. Thus, we can control the abstraction of the representation by this value, and adapt it to different applications and environments.

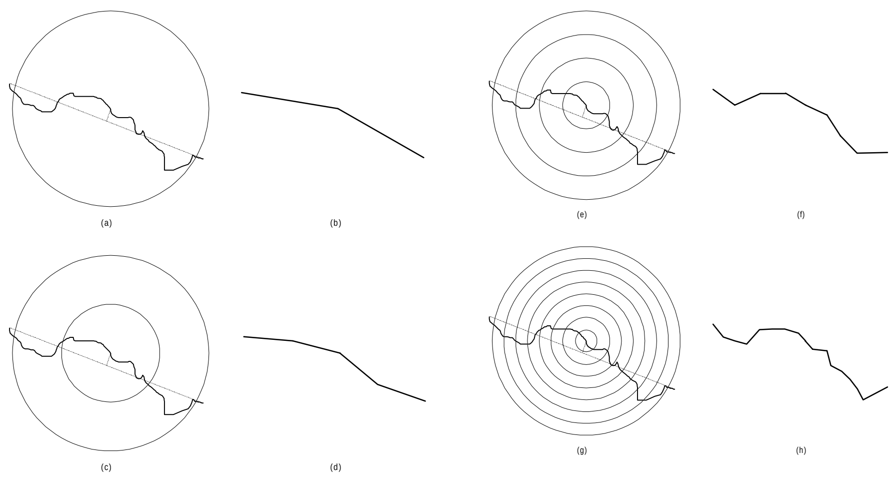


Figure 3: Hierarchical representation of a curve at different levels. (a) and (b) Original and reconstructed curves using one-circle representation, (c) and (d) using two-circle representation, (e) and (f) using four-circle representation, and (g) and (h) using eight-circle representation.

2.3 Matching

We say that some curve is a covered curve if and only if the following two conditions are satisfied. Firstly, it does not cross any representation circle at more than two points with the center point between them. Secondly, the curve's end points are at a distance less than or equal to twice the maximum radius of the representation circles. A covered curve has a single representation by the sets Φ and Ψ and it has one center point. When a curve is not a covered curve, then it will be divided into two or more covered subcurves or segments. We developed procedures to match two covered curves, and one covered curve with a general curve (possibly covered). We will start by discussing the first case and move towards the general one. The procedures are based on matching the representations of curves provided they were calculated using the same $r_{\min}$ and Δr parameters.

Given two curves C and D with representations Γ_C and Γ_D, respectively, both curves can be represented by information from an equal number of concentric circles, with

$$\Gamma_C = \{R_C, \Phi_C, \Psi_C\} \tag{2}$$

and

$$\Gamma_D = \{R_D, \Phi_D, \Psi_D\}. \tag{3}$$

For completeness, we include the radii of the circles at which each angle element was calculated. For example, ϕ_{3C} and ψ_{3C} are the representation angles calculated at radius $r_{3C} = r_{\min C} + (i-1)\Delta r$. This representation supports future extensions to cases where corresponding radii of different curves are not equal. The number of elements in each of these sets is M.

When the two curves match, the corresponding angles in the sets must be equal. First Φ_C is tested against Φ_D by comparing their corresponding elements, i.e. ϕ_{iC} is compared with ϕ_{iD} for $i = 1,\ldots,M$. If any pair is not equal, the match fails. This test is sufficient to detect match failure, but in order to conclude that the two curves match, one more test is required. First, we calculate another set of angles, Ξ. Each element ξ_i of this set represents the angle between the vector whose tip is at p_{i-1}^- and the one whose tip is at p_i^-, with radii r_{i-1} and r_i, respectively, and calculated as given by Equation (4) for $i = 2,\ldots,M$ and $\xi_1 = 0$.

$$\xi_i = \psi_i - (\phi_i - \phi_{i-1}) \tag{4}$$

Then, we compare the corresponding sum $\psi_i + \xi_i$ in both curves. This sum measures the total deviation of angles at radius r_i from those at radius r_{i-1}. Geometrically, this test is important to differentiate between curves that are symmetric about p_o.

It is preferred to start testing angles corresponding to circles with larger radii, and then the ones corresponding to smaller radii. The reason is that angles at very small radii are more likely to be equal even if the curves don't match. Thus, we will be able to decrease the time required to report a case where the matching fails.

A comparison that uses equality is appropriate in perfect situations. Practice suggests that many types of errors present in images of curves causes a test that is based on equality to fail, even if the two curves are identical in real world. A modification to this matching algorithm that is tolerant to errors is more robust. Instead of testing corresponding angles for equality, a relaxed condition would be to test if the absolute difference between them is less than some tolerance. In this case, a measure of representations' test is used. Equations (5) and (6) represent the mean square errors between the corresponding representation sets.

$$MSE_{CD}[\Phi] = \frac{1}{M}\sum_{i=1}^{M}\frac{r_i}{r_{\max}}\left(\phi_{iC} - \phi_{iD}\right)^2 \tag{5}$$

$$MSE_{CD}[\Psi + \Xi] = \frac{1}{M}\sum_{i=1}^{M}\frac{r_i}{r_{\max}}\left(\left(\psi_{iC} - \psi_{iD}\right) + \left(\xi_{iC} - \xi_{iD}\right)\right)^2 . \tag{6}$$

In our implementation, we chose a variable tolerance that is a function of the radius. Experiments showed that it is recommended to have tolerance values that get smaller as the radius increases. For both tests, we used functions of the form given by Equation (7) where K is a constant that can be different in both tests.

$$T(i) = K\frac{\Delta r}{r_{\min} + (i-1)\Delta r} \tag{7}$$

Note that the tolerance for the smallest circle ($i = 1$) is given by Equation (8) which is the largest value.

$$K\frac{\Delta r}{r_{\min}} \tag{8}$$

2.4 Performance

If the number of pixels on a curve is n, then preprocessing requires $O(n)$ computations. Another $O(n)$ computations are required to represent the curve using concentric circles information because every point on the curve trace is visited once to determine its distance from the center point. When matching is conducted on covered curves (curves represented by two sets of angles), the matching is $O(M)$-complex, where M is the number of circles used in the representation.

In some cases, the intention is to find a subcurve within a general curve. The subcurve can be represented by the two sets of angles, but the general curve cannot be represented. To enable such testing, the concentric circles are calculated for a subset of points from the general curve and tested against those calculated from the subcurve. This requires $O(nM)$ computations.

The number of circles, M, representing a curve plays a significant role in computations. A worst-case situation occurs when M is equal to $n/2$, i.e. the radius increment is very small that every point on the curve intersects a circle. In this case, the complexity is $O(n^2)$. On the average, M is much less than n.

3 Experimental Results

The algorithms discussed in the previous section were implemented using Khoros 2.0 under UNIX operating system on a SUN/SPARC machine.

Experiments were conducted on different classes of curves. We shall present curves from fingerprint images to demonstrate the effect of concentric circles curve matching. The notation fpx.y is used to represent fingerprint image y obtained from person x. Each fingerprint will generate few hundreds of curves whose representations become the database of known curves. Figure 4 shows four curves extracted from fingerprint fp7.1. Assume that this set of curves can uniquely identify the fingerprint. For illustration purposes, we marked the center of the concentric circles representation of each curve with a different number. The curves were matched against fingerprints fp1.3, fp2.4, and fp3.1 representing different individuals. They were also tested against another fingerprint (fp7.2) of the same individual. The matching results are shown in Figure 5. The matching curves are marked to identify the curve from Figure 4(b) against which they match. When a curve matches more than one curve from fp7.1, it is marked by (*).

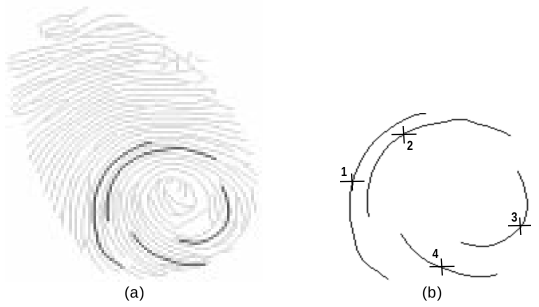


Figure 4: Fingerprint fp7.1. (a) Selected curves. (b) Center points marked.

A high-level procedure can analyze the arrangement of the marked centers of curves from fp7.1 to encode the pattern made by the curves. This pattern can be encoded by the following translation and rotation features:

1. The relative positions of each curve's center point to that of the other curves.
2. The relative orientation of each curve to the other curves.

This so-called pattern analysis procedure will test the results of curve matching to decide whether the same pattern exists. The inspection of the curves in Figure 5 shows that the matching curves of fp7.2 have a pattern similar to that of fingerprint fp7.1.

The previous discussion demonstrates one of many methods that can be used to study the patterns of the matching curves. A better approach to the same problem may assign weights to different curves based on their distinctiveness. Then, using the weights of curves and the matching patterns, a decision can be made to whether the two fingerprints belong to the same individual.

Other curve matching techniques can be used. The concentric circles technique has many features that can be utilized to speed up the fingerprint identification. Most curves extracted from fingerprints do not possess distinctive features. So feature-based approaches will not perform well here. Curve matching using concentric circles can be performed at high level of abstraction to select a set of candidate curves. Then, subsequent tests can be done on the set at lower levels of abstraction to fine-tune the match.

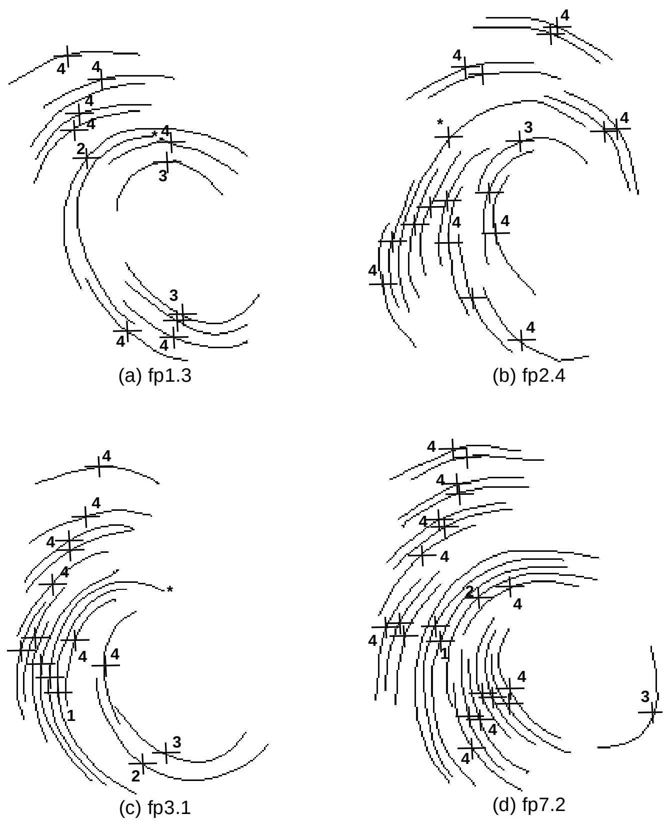


Figure 5: Curves matching the ones selected from fp7.1 in different fingerprints. (a) fp1.3. (b) fp2.4. (c) fp3.1. (d) fp7.2.

4 Conclusions and Future Work

In this research, concentric circles information was used to represent and match open curves. This representation is invariant to translation and rotation. The center of the circles is the point whose distances from the end points of the curve are equal. From this center, vectors with tips on the curve points that intersect the circles are constructed. The representation consists of two sets of angles calculated between vectors. The angles are indexed by the circles' radii. The parameters of the representation include minimum radius, radius increment, and maximum radius. The radius increment value is the radius difference between two consecutive circles and is constant for a representation. Matching examines the representations of different curves within some tolerance. The tolerance is a decreasing function of radius. This is justified by the fact that at higher radii, small changes in angles result in dramatic changes on the curves. In practice, tolerance is important to ensure the robustness of the matching test against noise and digitization errors.

The concentric circles approach has many advantages over the methods discussed in the introduction. In the template-based approach, a priori knowledge of the objects appearing in the application is essential for the calculation of salient features. Our approach does not require any prior knowledge of the curves. It does not require curves to possess any critical points. Feature-calculation approach uses critical points to segment curves. When curves are extracted from partially occluded objects, these critical points may or may not appear on the curves. The advantage of the feature-calculation approach is that the representation is invariant to scaling. Our algorithms are effective when used in a system with large number of curves to reject mismatches at early stages. The number of circles representing a curve determines the level of abstraction at which it is represented. The test can be conducted at high levels of abstraction and repeated again at lower levels for curves that passed the previous ones. Thus, mismatches are filtered as we proceed to lower levels of abstraction. We conducted tests on different classes of curves. The potential application on fingerprint recognition was discussed and showed promising results.

Concentric circles is a new approach. It can be improved to automate the selection of the radius increment value to enable curves to be represented at the highest possible abstraction. We are investigating building the representation as a hierarchical structure, and studying parallel algorithms for matching. In addition, we are looking into developing scale invariant sets from the concentric circles.

References

[1] H. J. Wolfson, “On curve matching,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-12, no. 5, pp. 483-489, May 1990.

[2] J. T. Schwartz and M. Sharir, “Identification of partially obscured objects in two and three dimensions by matching noisy characteristic curves,” *The International Journal of Robotics Research*, vol. 6, no. 2, pp. 29-44, 1987.

[3] J. Hong and H. J. Wolfson, “An improved model-based matching method using footprints,” in *Proceedings of the International Conference on Pattern Recognition*, Rome, Italy, pp. 72-78, November 1988.

[4] A. Kalvin, E. Schonberg, J. T. Schwartz, and M. Sharir, “Two-dimensional, model-based, boundary matching using footprints,” *The International Journal of Robotics Research*, vol. 5, no. 4, pp. 38-55, 1986.

[5] H. Freeman, “Shape description via the use of critical points,” *Pattern Recognition* vol. 10, pp. 159-166, 1978.

[6] P. Weiner, “Linear pattern matching algorithms,” in *Proceedings of the 14th Annual Symposium on Switching and Automata Theory*, IEEE Computer Society, pp. 1-11, 1973.

[7] J. W. Mckee and J. K. Aggarwal, “Computer recognition of partial views of curved objects,” *IEEE Transactions on Computers*, vol. C-26, no. 8, pp. 790-800, August 1977.

[8] P. L. Rosin, “Representing curves at their natural scales,” *Pattern Recognition*, vol. 25, no. 11, pp. 1315-1325, 1992.

[9] I. Weiss, “Noise-resistant invariants of curves,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 9, pp. 943-948, September 1993.

[10] H. Teh and R. T. Chin, “On the detection of dominant points on digital curves,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 8, pp. 859-872, August 1989.

[11] A. Pikaz and I. Dinstein, "Using simple decomposition for smoothing and feature point detection of noisy digital curves," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 8, pp. 808-813, August 1994.

[12] M. Worring and A. W. M. Smeulders, "Digital curvature estimation," *CVIP: Image Understanding*, vol. 58, no. 3, pp. 366-382, November 1993.

[13] H. Freeman and L. S. Davis, "A corner-finding algorithm for chain-coded curves," *IEEE Transactions on Computers*, pp. 297-303, March 1977.

[14] J. L. Turney, T. N. Mudge, and R. A. Volz, "Recognizing partially occluded parts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-7, no. 4, pp. 410-421, July 1985.