4th ICCV, Berlin, May, 1993

Silhouette-Based Object Recognition through Curvature Scale Space

Farzin Mokhtarian and Hiroshi Murase

NTT Basic Research Laboratories, Tokyo, Japan

Abstract

A complete and practical isolated-object recognition system has been developed which is very robust with respect to scale, position and orientation changes of the objects as well as noise and local deformations of shape (due to perspective projection, segmentation errors and non-rigid material used in some objects). The system has been tested on a wide variety of 3-D objects with different shapes and surface properties. A light-box setup is used to obtain silhouette images which are segmented to obtain the physical boundaries of the objects which are classified as either convex or concave. Convex curves are recognized using their four high-scale curvature extrema points. Curvature Scale Space Representations are computed for concave curves. The CSS representation is a multi-scale organization of the natural, invariant features of a curve. A three-stage, coarse-to-fine matching algorithm quickly detects the correct object in each case.

1. Introduction

Object representation and recognition is one of the central problems in computer vision. Normally, a reliable, working vision system must be able to **a)** effectively segment the image and **b)** recognize objects in the image using their representations. This paper describes a complete, working vision system which segments the image effectively using a light-box setup and recognizes isolated objects in the image reliably using their curvature scale space (or *CSS*) representations [4,5]. The CSS representation is based on the *scale space image* concept [7,10]. It is an organization of curvature zero-crossing points on a contour at multiple scales.

It is assumed that the recognition system developed here may be used for recognition of isolated 3-D objects. In particular, it is assumed that objects are placed one at a time on a light-box in front of a camera (by a robot arm, for example) and that the task is to recognize each object. We believe that this particular task is interesting for the following reasons:

a. Despite the constraints placed on the environment, *no* constraints have been placed on object shapes or types. Furthermore, environment constraints are not difficult to satisfy in many object recognition tasks (such as in industrial settings).

b. Every 3-D object, when placed on a flat surface and viewed by a fixed camera, has a limited number of stable positions, each of which can be modeled using a 2-D contour.

c. Even with only one object present on the light-box at a time, recognition can become challenging due to arbitrary shapes of objects, noise, and local deformations of shape which can be caused by perspective projection, segmentation errors and the non-rigid material used in some objects.

d. By considering only complete contour matching, we have developed a matching algorithm which we believe is optimal for that particular task.

The existing literature on shape representation and recognition is quite large. A survey of some recent work can be found in [8]. Linear features such as points, lines and planes were used in [1] for 3-D object matching. Sparse 3-D position and orientation measurements were used in the object recognition system described in [2]. A Hopfield neural network was used for object recognition in [6].

Sections 2 through 8 of this paper explain various aspects of the object recognition system that we developed. Section 2 explains how the segmentation of an image using a light-box system can be accomplished. Section 3 describes the CSS representation as a multi-scale organization of the inherent features of a planar curve. Section 4 shows how the maxima of a CSS representation can be extracted. Section 5 describes a fast CSS matching algorithm. Section 6 proposes a procedure for estimating the transformation parameters. Section 7 shows how the image-model curve distance can be computed. Section 8 describes a procedure for optimizing the transformation parameters. Section 9 gives an overall view of the implemented recognition system. Section 10 presents the results and an evaluation of the system. Section 11 contains the concluding remarks.

0-8186-3870-2/93 $3.00

2. Image Segmentation

The use of a light-box setup makes the segmentation of the image straightforward. The same threshold value (T = 120, with intensity values in the range: 0-255) was used to effectively segment all input images including those of objects made of colored transparent material (tape dispenser and screw driver). The output of the thresholding step is a binary image to which a salt-and-pepper noise removal procedure was applied. The resulting binary image always had only one connected region which corresponded to the object. The bounding contour of that region was then recovered (figure 10.2).

3. The curvature scale space representation

A CSS representation is a multi-scale organization of the invariant geometric features (curvature zero-crossing points and/or extrema) of a planar curve (here, only curvature zero-crossings were used). The CSS representation of a planar curve Γ represents that curve uniquely modulo scaling and a rigid motion [3]. To compute it, Γ is first parametrized by the arc length parameter u:

$$\Gamma(u) = (x(u), y(u)).$$

It is assumed that the input curve is initially represented by a polygon with possibly many vertices. Therefore only the coordinates of the vertices of the polygon need be given. If all the distances between adjacent vertices of the polygon are equal, then an arc length parametrization of the curve is already available. Otherwise, that polygon is sampled to obtain a new list of points such that all the distances between points adjacent on the list are equal *on the original polygon*. An *evolved version* Γ_σ of Γ can then be computed. Γ_σ is defined by:

$$\Gamma_\sigma = (X(u,\sigma), Y(u,\sigma))$$

where

$$X(u,\sigma) = x(u) \circledast g(u,\sigma)$$
$$Y(u,\sigma) = y(u) \circledast g(u,\sigma)$$

where $\circledast$ is the convolution operator and $g(u,\sigma)$ denotes a Gaussian of width σ. The process of generating evolved versions of Γ as σ increases from 0 to ∞ is referred to as the *evolution* of Γ. The CSS representation contains curvature zero-crossings or extrema extracted from evolved versions of the input curve. In order to find such points, we need to compute curvature accurately and directly on an evolved version Γ_σ of a planar curve. It can be shown [5] that curvature κ on Γ_σ is given by:

$$\kappa(u,\sigma) = \frac{X_u(u,\sigma)\, Y_{uu}(u,\sigma) - X_{uu}(u,\sigma)\, Y_u(u,\sigma)}{(X_u(u,\sigma)^2 + Y_u(u,\sigma)^2)^{1.5}}$$

where

$$X_u(u,\sigma) = \frac{\partial}{\partial u}(x(u) \circledast g(u,\sigma)) = x(u) \circledast g_u(u,\sigma)$$

$$X_{uu}(u,\sigma) = \frac{\partial^2}{\partial u^2}(x(u) \circledast g(u,\sigma)) = x(u) \circledast g_{uu}(u,\sigma)$$

and $Y_u(u,\sigma)$ and $Y_{uu}(u,\sigma)$ are defined similarly. The CSS image of Γ is defined as the solution to $\kappa(u,\sigma) = 0$. Note the following:

a. The CSS image is stored as a binary image in which each row corresponds to a specific value of σ and each column corresponds to a specific value of u.
b. A brute force computation of a CSS image will, in general, require the evaluation of a large number of convolutions which can slow the system down. The method used here was to *track* the zero-crossings in the CSS image: at each scale during computation, curvature is computed only in a small neighborhood of each location where a zero-crossing was detected at the previous scale. This is possible since for a small change in σ, the change in location of any curvature zero-crossing point on the curve is also small.
c. For all values of σ larger than a σ_c, evolved curves Γ_σ will be simple and convex. Hence computation can stop as soon as σ_c is reached or as soon as no more curvature zero-crossings are detected on Γ_σ.

4. Extracting maxima of CSS contours

As described in the next section, the features of the CSS image used for matching are the maxima of the CSS contours. These maxima are not readily available and must be extracted from the CSS image. As seen in figures 10.3(b) and 10.4(b), a CSS contour is usually connected everywhere except in a neighborhood of its maximum. Near the maximum of a CSS contour, the slope is very close to zero. As a result, even with fine sampling of the input curve and the σ parameter, it is unlikely that CSS contours will be closed at their maxima. (Furthermore, very fine sampling will result in a large CSS image and computational cost.) So the actual maximum of a CSS contour usually falls in the *gap* at the top of that contour and it is assumed that each contour consists of two disconnected segments. The top point of each segment is located and the peak is considered to be the midpoint of the line segment connecting the top points.

5. Curvature scale space matching

The basic idea behind the CSS matching algorithm is to obtain a coarse-level match using the structural features of the input curves. Such a match can be found quickly and reliably since at the high levels of CSS images, there are relatively few features to be matched. The actual features used for matching are the maxima of the curvature zero-crossing contours since they are the

most significant points of those contours: the CSS coordinates of a maximum convey information on both the location and the scale of the corresponding contour whereas the *body* of the contour is, in general, similar in shape to those of other contours. Furthermore, the maxima are isolated point features and therefore solving the feature correspondence problem is relatively simple. This is specially true at the high scales of the CSS image where the maxima are sparse.

So the task of the matching algorithm is to find the correct correspondence between two sets of maxima: one from each CSS image. The allowed transformation from one set to the other is mere horizontal translation (since all contours have the same number of sampled points). The translation parameter is computed when the first image curve CSS maximum is mapped to the first model curve CSS maximum and then used to map each of the remaining image curve CSS maxima to the model curve CSS. The corresponding model curve CSS maximum for each mapped image curve CSS maximum should then be the *closest* model curve CSS maximum (and the associated cost is the Euclidean distance between them). Many candidates may have to be considered since the correspondence between the first pair of maxima can be made in possibly many ways. This matching problem is solved using a *best-first* matching strategy [9] which will gradually expand a number of candidate matches in parallel (always selecting the best partial match) until the lowest-cost complete match is found.

6. Estimating the transformation parameters

Once the best match of two CSS representations has been determined, it is possible to compute an initial approximation for the transformation parameters since the correspondence between arc length values on the two curves is known. It is assumed that the transformation to be solved for consists of uniform scaling, rotation and translation in x and y. Let

$$\mathbf{X} = (x_j, y_j)$$

be a set of η points on the model curve and let

$$\Xi = (\xi_j, \psi_j)$$

be the set of corresponding points on the image curve. The parameters of the following transformation:

$$\begin{aligned} x_j &= a\,\xi_j + b\,\psi_j + c \\ y_j &= -b\,\xi_j + a\,\psi_j + d \end{aligned} \qquad (6.1)$$

must be solved for. A *Least-Squares Estimation* method is used to estimate values of a, b, c and d. Let the *dissimilarity measure* Ω which measures the difference between the model curve and the transformed curve be defined by:

$$\Omega = \sum_{j=1}^{\eta} (x_j - x_j^c)^2 + (y_j - y_j^c)^2$$

where (x_j^c, y_j^c) is a point on the model curve corresponding to transformed image curve point (x_j, y_j). Using equation (6.1) to eliminate x_j and y_j yields:

$$\Omega = \sum_{j=1}^{\eta} (a\xi_j + b\psi_j + c - x_j^c)^2 + (-b\xi_j + a\psi_j + d - y_j^c)^2.$$

Let $\mathbf{P} = (a, b, c, d)$ be the vector defined by the transformation parameters. The solution of

$$\partial\Omega / \partial\mathbf{P} = 0$$

is the least-squares estimate of those parameters. Compute $\partial\Omega / \partial a$ and set it to zero:

$$\frac{\partial\Omega}{\partial a} = \sum_{j=1}^{\eta} 2a\xi_j^2 + 2c\xi_j - 2\xi_j\xi_j^c + 2a\psi_j^2 + 2d\psi_j - 2\psi_j y_j^c = 0.$$

Repeat for the other transformation parameters. The result is a linear system of four equations in four unknowns which is solved to obtain the least-squares estimate of a, b, c and d.

7. Measuring image-model curve distances

Once an estimate of the transformation parameters is available, it is possible to map the image curve to the space of the model curve. It is then useful to measure the image-model curve distance for two reasons:

a. As described in section 9, sometimes the image curve is mapped to different model curves in order to determine which model curve is closest to the image curve. This is accomplished by measuring image-model curve distances.

b. The computation of the image-model curve distance is essential to transformation parameter optimization as described in section 8.

To compute the image-model curve distance, the following procedure is carried out for each vertex of the image curve: the closest vertex v_i of the model curve is located. Vertex v_i and its two neighboring vertices v_{i-1} and v_{i+1} are then used to locate the closest *point* of the model curve. The distance to that point is then computed. The image-model curve distance is the sum of the individual distances.

8. Optimizing the transformation parameters

The least-squares estimate of the transformation parameters computed in section 6 is, in general, *not* the optimal estimate. This is because the image-model point correspondences computed from the CSS match are not precise due to noise and local shape distortions. Nevertheless, it is possible to optimize those parameters using the following procedure:

a. Compute the least-squares estimate of the parameters using the technique described in section 6 and use it to map the image curve to the model curve.

b. Compute the image-model curve distance and determine a new set of corresponding points on the model curve as described in section 7.
c. If the difference between the last value of the image-model curve distance and its previous value is less than $\varepsilon > 0$, then STOP, otherwise, go to step **a**.

In our system, it was possible to compute the optimal parameters with less than 1% error using at most 10 iterations of the procedure described above.

9. A silhouette-based recognition system

It is now possible to give an overall description of the system designed and implemented for silhouette-based object recognition through the CSS representation. The system was designed so that both *convex* and *concave* curves can be recognized. It can be divided into an *off-line* and an *on-line* component. The off-line part is completed first and is itself divided into two stages: *model acquisition* and *computing model representations*. Note that a resolution of 200 sampled points was used for all model and image contours.

a. Model Acquisition: One contour must be obtained for each stable position of each model object. Model contours are rescaled so that they all *just* fit inside the same square.

b. Computing Model Representations: If a model contour is concave, its CSS representation is computed as described in section 3. The aspect ratio (ratio of the number of rows to the number of columns) of that CSS image is also computed. The maxima M_k of the CSS contours are located (section 4). The final representation for the model contour is simply the CSS coordinates of each M_k. If a model contour is convex, that contour is smoothed until only 4 curvature extrema remain on the contour. The arc length coordinates of those 4 extrema are then recorded. The ratio of the Euclidean distance between the curvature maxima to the distance between the curvature minima is also computed for each convex contour.

The on-line part of the system can be divided into several stages as following:

a. The input image is first segmented using the procedure described in section 2 to obtain a contour from the image. The contour is smoothed slightly to remove noise and normalized to fit in a square of size one with its lower left corner at the origin. Curvature is then computed at each point along the contour. If curvature is nearly constant, the system concludes that the contour is circular and stops. Otherwise, if curvature is positive everywhere or very close to zero but possibly negative (this situation can be caused by noise on a convex contour), the system concludes that the input contour is convex and follows steps **g** through **j**. Otherwise, the input contour is considered to be concave and steps **b** through **f** are followed.
b. The CSS representation of image curve is computed.
c. The maxima of image curve CSS contours are located.
d. The aspect ratio of the image curve CSS is computed. Any model curve CSS image whose aspect ratio is *close* (see section 10) is accepted for step **e**.
e. CSS matching is applied to the surviving models.
f. The *best* (see section 10) matches in step **e** are selected for verification. For each selected model curve, image curve transformation parameters are computed using their best CSS match (section 6) and used to map the image curve to that model. The image-model curve distance is then computed (section 7). The model curve with the lowest image-model curve distance is chosen as the best matching model. Transformation parameter optimization is then used to find the best fit of the image curve to the chosen model. In situations where there is little difference between some model curves, parameter optimization can be integrated into the recognition process.
g. The input convex contour is smoothed until only 4 curvature extrema remain on the contour.
h. The ratio of the Euclidean distance between the curvature maxima on the contour to the distance between the curvature minima on the contour is computed. Any convex model curve whose corresponding ratio is *close* (see section 10) is accepted for step **i**.
i. The first curvature maximum on the image curve can map to either of the two curvature maxima on each of the surviving model curves. A set of image curve transformation parameters is computed for each case and applied to the image curve. The image-model curve distance is computed in each case.
j. The model curve with the lowest image-model curve distance is chosen as the best matching model. Parameter optimization is then used to find the best fit of the image curve to the chosen model.

10. Results and discussion

The recognition system described in section 9 was implemented in *C* and ran on a *SiliconGraphics IRIS 4D/85GT* workstation. It was tested using a total of 22 model curves and 19 images. One circular model contour (a *spray can lid*) and two convex model contours (a *calculator* and a *glue stick*) were used. All other model contours (a *bottle*, a *paper clip*, a *fork*, a *key*, a *monkey wrench* (two sides), a *panda*, two *connector cases*, a *screw driver*, a pair of *scissors*, a *spoon*, a *tape dispenser*, a *vase*, two *wrenches* (two sides each) and a *wire cutter*) were concave. A number of the model con-

tours used to test the system are shown in figure **10.1**.

Due to the light-box setup used, thresholding was successful in properly segmenting each input image and recovering the bounding contours (figure **10.2**).

Each one of the 19 input objects was recognized correctly by the system in less than 2 seconds (this includes time for computation of image curve CSS representation). In most cases, CSS matching was sufficient to correctly recognize the input object. When ambiguities remained after CSS matching (for example, when two or more objects had the same coarse-level shape structure), curve distance computation successfully resolved those ambiguities.

We believe the matching program is fast because it has been designed specifically for the task of matching closed curves using maxima of CSS contours. These maxima (excluding the ones corresponding to very small CSS contours) are usually small in number (from 2 to 6 for the input curves) but have a high multi-scale discriminative power. As a result, we believe they are the ideal feature points for the matching task considered here. Furthermore, our coarse-to-fine matching technique makes efficient use of recognition power: models pass through *recognition filters* which become increasingly fine until the best-matching model is selected.

We drastically changed the thresholds used in the tests which act as filters between the various stages of the system (see steps **d**, **f** and **h** in section 9). In one test, those filters were effectively removed. It was verified that there were no effects on the output of the system; the filters exist only for efficiency reasons. The system was very robust in each case despite the presence of noise and local deformations of shape due to:

a. Perspective projection of actual 3-D objects with considerable depth (such as the vase).
b. Segmentation errors near smooth physical boundaries.
c. Non-rigid material (such as cloth or soft plastic) used in some input objects.

It was also discovered that a single model can be used to represent a class of similar looking objects. For example, the model screw driver was slightly different from the screw driver used as input to the system.

The following are some examples of the matches found by the system. In each case, the image curve (drawn using a thin line) has been mapped to the model curve (drawn using a thick line). Figure **10.3** shows the Panda matched to its model. Note that the Panda was made of cloth and therefore did not have a very rigid shape. Figure **10.4** shows the vase matched to the model vase. The local mismatch that can be observed is due to the fact that the model curve corresponds to an orthogonal projection of the vase whereas the image curve corresponds to its perspective projection.

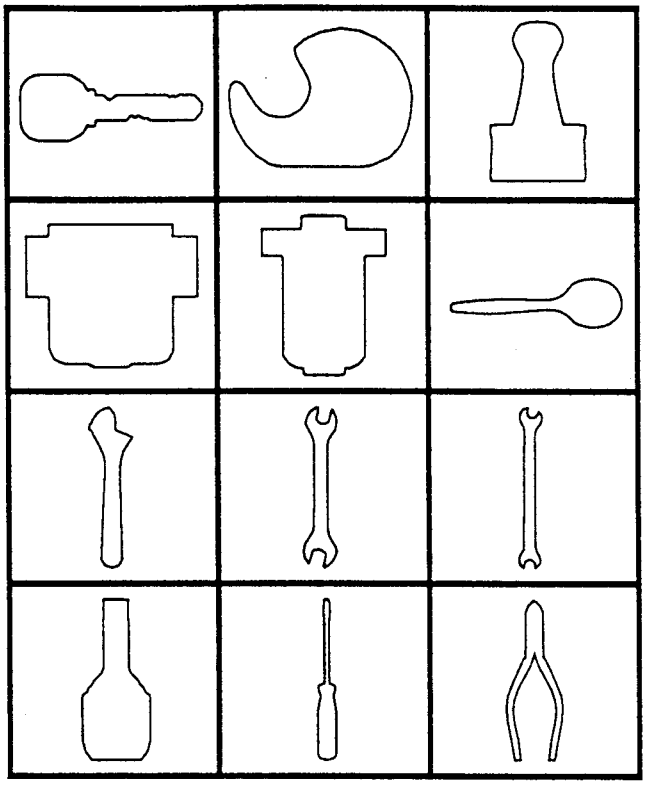
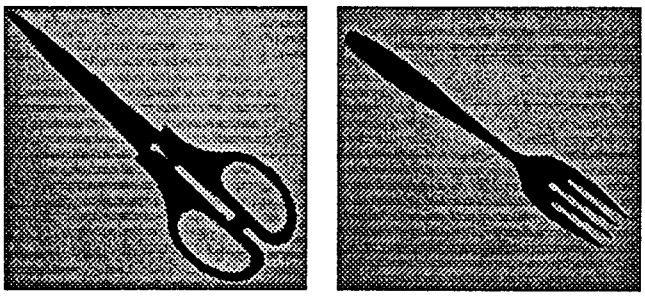
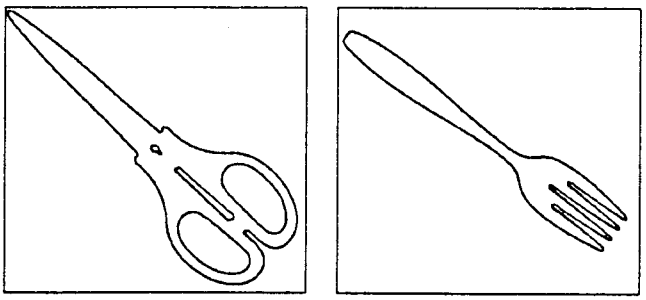


Figure 10.1. A number of model contours

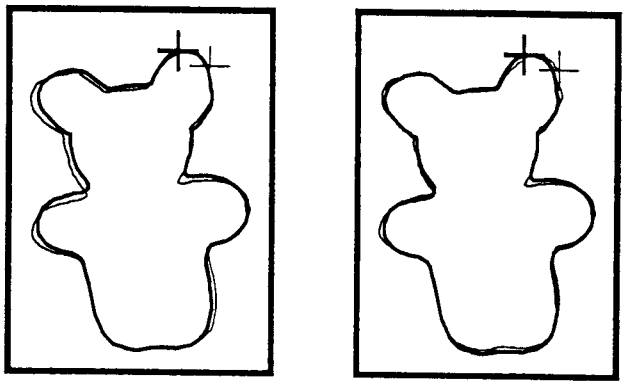


a. Two images taken using the light-box

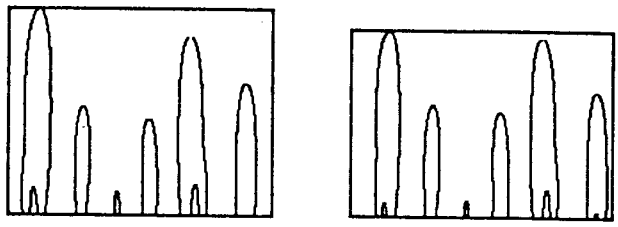


b. Contours recovered from images in (a)

Figure 10.2

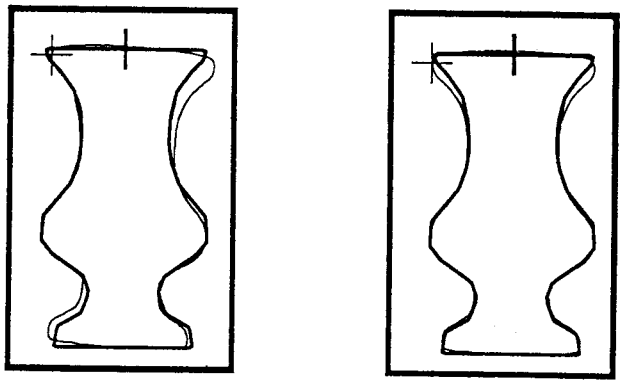


a. Panda match before transformation parameter optimization (left) and after (right)

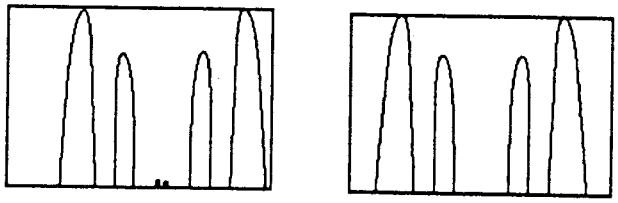


b. CSS of image (left) and model curves (right). CSS origins are marked on contours in (a).

Figure 10.3



a. Vase match before transformation parameter optimization (left) and after (right)



b. CSS of image (left) and model curves (right). CSS origins are marked on contours in (a).

Figure 10.4

11. Conclusions

This paper described a complete and practical isolated-object recognition system which used a light-box setup to obtain silhouette images of objects and recover their bounding contours. Those boundaries were classified as either convex or concave. Convex curves were recognized using their four high-scale curvature extrema points. CSS representations were computed for concave curves. A three-stage, coarse-to-fine matching algorithm (consisting of the CSS aspect ratio test, CSS matching and image-model curve distance computation) was used to find the correct model for each concave image curve. Transformation parameter optimization was then used to find the best fit. The system was tested on a variety of 3-D objects with different shapes and surface properties. It was found to be very robust with respect to position, orientation and scale changes of the objects as well as noise and local shape distortions.

12. Acknowledgments

This work was supported by NTT Basic Research Laboratories. We are grateful to Seiichiro Naito and John Canning for helpful discussions.

References

[1] Faugeras, O. D. and M. Hebert, "The Representation, Recognition, and Locating of 3-D Objects," *Int. J. Robotics Research*, vol. 5, no. 3, pp. 27-52, 1986.

[2] Grimson, W. and T. Lozano-Perez, "Model-Based Recognition and Localization from Sparse Range or Tactile Data," *Int. J. Robotics Research*, vol. 3, no. 3, pp. 3-35, 1984.

[3] Mokhtarian, F., "Fingerprint theorems for curvature and torsion zero-crossings," *Proc. IEEE Conf. CVPR*, pp. 269-275, San Diego, California, 1989.

[4] Mokhtarian, F. and A. K. Mackworth, "Scale-Based Description and Recognition of Planar Curves and Two-Dimensional Shapes," *IEEE Trans. PAMI*, vol. 8, no. 1, pp. 34-43, 1986.

[5] Mokhtarian, F. and A. K. Mackworth, "A Theory of Multi-Scale, Curvature-Based Shape Representation for Planar Curves," *IEEE Trans. PAMI*, vol. 14, no. 8, pp. 789-805, 1992.

[6] Nasrabadi, N. M., W. Li and C. Y. Choo, "Object Recognition by a Hopfield Neural Network," *Proc. Int. Conf. Computer Vision*, pp. 325-328, 1990.

[7] Stansfield, J. L., "Conclusions from the Commodity Expert Project," memo 601, MIT AI Lab, Cambridge, MA, 1980.

[8] Stein, F. and G. Medioni, "Structural Indexing: Efficient 3-D Object Recognition," *IEEE Trans. PAMI*, vol. 14, pp. 125-145, 1992.

[9] Winston, P. H., *Artificial Intelligence*, Addison-Wesley, Reading, MA, 1979.

[10] Witkin, A. P., "Scale Space Filtering," *Proc. IJCAI*, pp. 1019-1023, Karlsruhe, Germany, 1983.