Silhouette-Based Object Recognition with Occlusion through Curvature Scale Space

Farzin Mokhtarian

Department of Electronic and Electrical Engineering
University of Surrey, Guildford, England GU2 5XH
E-mail: *F.Mokhtarian@ee.surrey.ac.uk*

Abstract. A complete and practical system for object recognition with occlusion has been developed which is very robust with respect to noise and local deformations of shape as well as scale changes and rigid motions of the objects. The system has been tested on a wide variety of 3-D objects with different shapes and surface properties. No restrictive assumptions have been made about the shapes of admissible objects. An industrial application with a controlled environment is envisaged. The *Curvature Scale Space* technique [4, 5] is used to obtain a novel *multi-scale segmentation* of the image contour and the model contours using curvature zero-crossing points. Multi-scale segmentation renders the system substantially more robust with respect to noise and local shape differences. *Object indexing* [9] is used to narrow down the search-space and avoid an exhaustive investigation of all model segments. A local matching algorithm applies *candidate generation, selection, merging, extension* and *grouping* to select the best matching models.

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. A complete and robust isolated object recognition system was described in [6, 3]. This paper describes a complete, working vision system which segments the image effectively using a light-box setup and recognizes occluded 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 proposed in [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 occluded 3-D objects. In particular, it is assumed that a number of objects are placed on a light-box directly in front of a camera and that the task is to recognize each object. We believe that this particular task is interesting for the following reasons:

- 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 recognition tasks (such as industrial settings).

- Every 3-D object resting on a flat surface and viewed by a fixed camera, has a few stable positions, each of which can be modeled by a 2-D contour.
- Recognition can still become challenging due to arbitrary shapes of objects, noise, and local deformations of shape which can be caused by perspective projection, segmentation errors and non-rigid material.

The existing literature on shape representation and recognition is quite large. A survey of some recent work can be found in [9]. It should be noted that projective invariants [8] have received attention recently as tools for object recognition.

In general, a shortcoming of some object representation techniques is that the features extracted from the objects and used for matching are too local and therefore the resulting system is not robust with respect to noise and local deformations of shape. In those systems in which less local features are used, the utilized features are not necessarily inherent features of the object and therefore have weak discriminative power.

Sections 2 through 8 of this paper explain various aspects of the object recognition system developed. Section 2 briefly explains how the segmentation of an image using a light-box system is accomplished. Section 3 reviews the CSS representation as a multi-scale organization of the inherent features of a planar curve. Section 4 shows how multi-scale segmentation of a 2-D contour using curvature zero-crossing points may be accomplished. Section 5 describes a fast, local 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 efficient optimization of the transformation parameters. Section 9 presents the results and an evaluation of the system. Section 10 contains the concluding remarks.

2 Image Segmentation

The use of a light-box setup makes the segmentation of the image reasonably straightforward. The same threshold value T was used to effectively segment all input images. A salt-and-pepper noise removal procedure was applied to the resulting binary image in order to remove isolated noise. The next step is to apply a process of region growing followed by shrinking to the image in order to fill in cracks and small holes. The resulting binary image always had only one connected region of 1-pixels which corresponded to the objects. Next, boundary pixels belonging to the region of 1-pixels are detected. The final step is to recover the image coordinates of the boundary points.

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 represents that curve uniquely modulo scaling and a rigid motion [2]. To compute it, the curve 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 the distances between adjacent vertices of the polygon are all 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 the distances between points adjacent on the list are all equal *on the original polygon.* An *evolved version* Γ_σ of Γ can then be computed. Γ_σ is defined by [4]:

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

where

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

where $\otimes$ 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. This technique is suitable for removing noise from a planar curve. Evolving contours can be considered an early form of active contours (snakes) [1] since they are similar in behaviour to snakes without any external constraints.

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 that curvature κ on Γ_σ is given by [5]:

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

where

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

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

and

$$Y_u(u,\sigma) = y(u) \otimes g_u(u,\sigma) \qquad\qquad Y_{uu}(u,\sigma) = y(u) \otimes g_{uu}(u,\sigma).$$

The function defined implicitly by $\kappa(u,\sigma) = 0$ is the CSS image of Γ. Note that:

- The CSS image is stored as a binary image in which each row corresponds to a specific value of σ and each column to a specific value of u.

- A brute force computation of a CSS image can be inefficient. The method usually used is to *track* the zero-crossings in the CSS image: at each scale 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 on the curve is also small.
- For all values of σ larger than a σ_c, evolved curves Γ_σ will be simple and convex. This suggests that the computation can stop as soon as σ_c is reached or as soon as no more curvature zero-crossings are detected on Γ_σ [7].

For examples of CSS images, see [5, 6].

4 Multi-Scale Segmentation of 2-D Contours

The basic idea behind the segmentation scheme is to divide the input contour into primitive segments to be used by a local matching algorithm (described in section 5). Curvature zero-crossing points are the natural feature points to divide the contour since their locations are invariant with respect to rotation, scaling and translation of the contour. The main issue, therefore, is the issue of scale: *which scale should be chosen for the detection of curvature zero-crossing points on the input contour?* If the scale chosen is too small, the segmentation may be affected by noise and local distortions of shape, and if it is too large, important structure on the contour may be lost.

The solution used here was motivated by the main underlying concept of the curvature scale space representation: *utilize information from multiple scales rather than prefer a single scale.* Therefore the segmentation of the input contour is also carried out at multiple scales. The procedure is as follows:

- Start the segmentation at the lowest scale of the CSS image and end at a medium scale since the segments discovered at high scales are not useful.
- Segment the contour using the curvature zero-crossing points detected at the lowest scale and add all segments (defined by their left and right endpoints expressed in arc-length values) to a segment-list. As each higher scale is considered, again detect all curvature zero-crossing points at that scale *but* add a new segment if it does not already exist in the segment-list.

Care must be taken to account for the movement of curvature zero-crossing points in the CSS image. Therefore an auxiliary segment-list is also used which always records the updated values (across scales) of the left and right endpoints of each segment in the original segment-list. To check for existence, the auxiliary segment-list is searched. When extracted segments are written to the output file, the original segment-list is used since the segments in the auxiliary segment-list become very small at the maximum of the corresponding CSS zero-crossing contour. This multi-scale segmentation scheme is substantially more robust with respect to noise and local shape differences.

5 Local Matching through Curvature Scale Space

Due to occlusion, the matching algorithm employed is a local one and consists of several stages. This section describes those stages in the sequence in which they are carried out.

5.1 Rescaling

Model contours and the image contour are rescaled so that they just fit in a unit square. The model contours are further rescaled so that they reflect the relative sizes of model objects when viewed at the same distance. Model contour rescaling is carried out off-line. As a result of image and model contour rescaling, the possible scale changes from model contours to the image contour become predictable which helps to define an *admissible space* for the scale-factors (In principle, since the distance from the camera to the light-box is known, the scale-factors are also known, but that information was not used here: the system is therefore allowed to recover the correct scale-factors as a result of the recognition process). The multi-scale segmentation procedure described in section 4 is then used to segment the model contours and the image contour.

5.2 Candidate Generation and Filtering

Due to occlusion, all possible local matches must be considered (note however, that very small segments on either the image contour or the model contours are discarded). In order to avoid an exhaustive search of all model contour segments, *object indexing* [9] is employed to render the initial search more efficient. After segmentation, each model contour segment is rescaled so that each has the same length $\mathcal{L}$ (subject to constraints imposed by the admissible space defined in the previous subsection). Average curvature is then computed for each of those segments and used to create an index-table for all the model contour segments. All the computation is carried out off-line.

Once the segmentation of the image contour is completed, each image contour segment is also rescaled so that each has the same length $\mathcal{L}$ (again subject to constraints imposed by the admissible space). Average curvature is also computed for each of those segments. The average curvatures now serve as indices into the model contour segment index-table to recover a more likely (and smaller) set of potentially matching model contour segments. A candidate is generated for the possible match of each image contour segment and the corresponding model contour segments recovered from the index-table.

Transformation parameter optimization is then applied (as described in section 8) to the generated candidates in order to refine the initial estimate of those parameters. *This step is crucial since the accuracy of segment distance calculation depends greatly on the accuracy of the transformation parameters.* For each candidate, *segment-dist* is defined as the average point distance between the image-model contour segments (see section 7) and used as a measure of the goodness of fit between the two segments. A number of candidates with low *segment-dist* values are then selected for further processing.

5.3 Candidate Merging

Initial candidates correspond to simple segments delimited by neighboring curvature zero-crossing points. Nevertheless, it is possible for the visible boundary of an object in the input image to be divided into several neighboring or even overlapping segments. It is therefore necessary to merge those initial candidates which satisfy several criteria intended to measure candidate compatibility. It follows that two candidates c_1 and c_2 will be merged if they satisfy the following criteria:

- c_1 and c_2 must be valid (not previously merged) and different candidates.
- c_1 and c_2 must correspond to the same model.
- The transformation parameters of c_1 and c_2 should be *roughly* the same.
- The corresponding segments of c_1 and c_2 must be neighboring or overlapping.
- The scale factor associated with the new candidate must be admissible.
- The new candidate must have a low *segment-dist* value.

When two candidates are merged, the corresponding segments will be the union of the old segments. The old candidates are invalidated. Candidate merging will continue until no two candidates can be found which satisfy the merging criteria.

5.4 Candidate Extension

In general, the intersection point of two object boundaries in the input image does not coincide with an endpoint of a curvature zero-crossing segment. Therefore in order to find the exact location of such intersection points, it is necessary to gradually extend the contour segments associated with the merged candidates as long as a good fit between the image and model segments can be observed. Extension is first carried out at the right endpoint until mismatch error is too large and then carried out at the left endpoint. It is assumed that in general, object intersection points are a subset of the curvature maxima on the image contour (this is true except in hypothetical situations). First, all curvature extrema are located on a slightly smoothed version of the input image contour. Then, the following procedure is applied at each endpoint of each candidate:

- Extend the image contour segment to the next curvature maximum.
- The corresponding model contour segment is extended accordingly.
- Determine new transformation parameters and the new value of *segment-dist* for the candidate being extended.
- Determine the number of points k in a small neighborhood of the endpoint which are far from the image contour.

Extension stops if either the new candidate no longer has a low *segment-dist* value, or the new value of *segment-dist* rises sharply compared to previous value, or k rises above an acceptable limit. When extension stops, tests are carried out to detect a borderline case (k is just above the acceptable limit or value of *segment-dist* is just above the cut-off threshold). If so, the current endpoint becomes the final endpoint. Otherwise, the previous endpoint becomes the final endpoint.

5.5 Candidate Grouping

The next step in matching is to group compatible but disjoint candidates. The tests applied to determine compatibility are the same as the tests in section 5.3 except that the fourth test is not applied. It is certainly possible that, due to occlusion, an object in the scene may appear as two or more disjoint components in the input image. The goal of this step is to identify such situations to aid in the process of recognition.

5.6 Candidate Selection

What remains is to select the *best* candidates using an appropriate criterion. As stated earlier the value of *segment-dist* for each candidate is the average point distance between the contour segments associated with that candidate. This is a suitable measure of how well the shapes of those contour segments match. Another measure of the significance of a candidate is its *support*. Candidate support is defined as the length of the image contour segment associated with the candidate (note that if two disjoint candidates are found in section 5.5 to be compatible, the support of each candidate is increased by the length of the image contour segment associated with the other candidate). Define:

$$candidate-cost = \frac{segment-dist}{candidate-support}.$$

Note that a candidate with a lower cost is a *better* candidate. The following procedure is then used to select the best candidates:

- Determine the cost of each candidate.
- Select the valid candidate with the lowest cost.
- Disqualify all candidates whose corresponding image contour segment overlaps with the image contour segment of the chosen candidate or the image contour segment of any candidate compatible (see section 5.5) with the chosen candidate.
- Find any image contour segments delimited by negative curvature minima which do not overlap with the image contour segments associated with any chosen candidates or candidates compatible with them, and which fit well with the model associated with the chosen candidate. Examples are straight line segments which do not occur in valid candidates.
- Disqualify all candidates whose corresponding image contour segment overlaps with any of the image contour segments discovered in the previous step.
- Determine the final fit of the model associated with the chosen candidates using all relevant image contour segments and map the model to the image space.

- Disqualify the chosen candidate and all candidates compatible with it.
- If any valid candidates remain, go to the second step above, otherwise STOP.

Note that this procedure is independent of number of objects in input image.

6 Solving for the Transformation Parameters

When mapping a model curve segment to an image curve segment, it is possible to obtain many pairs of points on those segments in order to compute an initial approximation for the transformation parameters since the correspondence between arc length values on the curve segments is known. It is assumed that the transformation to be solved for consists of uniform scaling, rotation and translation in x and y. Let $\mathcal{X} = (x_j, y_j)$ be a set of η points on the image curve and let $\Xi = (\xi_j, \psi_j)$ be the set of corresponding points on the model curve. The parameters of the following transformation:

$$x_j = a\xi_j + b\psi_j + c \qquad y_j = -b\xi_j + a\psi_j + d \tag{1}$$

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 segment and the image curve segment be defined by:

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

where (x_j^c, y_j^c) is the closest point on the image curve to transformed model curve point (x_j^t, y_j^t). Using equation (1) to eliminate x_j^t and y_j^t 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 $\mathcal{P} = (a, b, c, d)$ be the vector defined by the transformation parameters. The solution of $\frac{\partial \Omega}{\partial \mathcal{P}} = 0$ is the least-squares estimate of those parameters. To compute that estimate, determine the partial derivatives of Ω with respect to each of a, b, c and d and set those partial derivatives to zero. The result is a linear system of four equations in four unknowns which is solved to obtain estimates for a, b, c and d:

$$a = \frac{\sum \xi_j x_j^c + \sum \psi_j y_j^c - \frac{1}{\eta}\sum x_j^c \sum \xi_j - \frac{1}{\eta}\sum y_j^c \sum \psi_j}{\sum \xi_j^2 + \sum \psi_j^2 - \frac{1}{\eta}\sum \xi_j \sum \xi_j - \frac{1}{\eta}\sum \psi_j \sum \psi_j}$$

$$b = \frac{\sum \psi_j x_j^c - \sum \xi_j y_j^c + \frac{1}{\eta}\sum y_j^c \sum \xi_j - \frac{1}{\eta}\sum x_j^c \sum \psi_j}{\sum \xi_j^2 + \sum \psi_j^2 - \frac{1}{\eta}\sum \xi_j \sum \xi_j - \frac{1}{\eta}\sum \psi_j \sum \psi_j}$$

$$c = \frac{\sum x_j^c - a\sum \xi_j - b\sum \psi_j}{\eta}$$

$$d = \frac{\sum y_j^c + b\sum \xi_j - a\sum \psi_j}{\eta}.$$

7 Measuring Image-Model Curve Distances

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

- Different model curves are mapped to the image curve in order to determine which model curve is locally closest to the image curve. This is accomplished by measuring image-model curve segment distances.
- The computation of the image-model curve segment distance is essential to transformation parameter optimization as described in section 8.

The image-model curve segment distance is computed by determining the closest point on the image curve segment (not necessarily a vertex) to each vertex of the model curve segment, and averaging the corresponding 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 are not precise due to noise and local shape distortions. Nevertheless, it is possible to optimize those parameters as following:

- Let $D_p = \infty$.
- Compute the least-squares estimate of the parameters using the technique described in section 6 and use it to map the model curve to the image curve.
- Determine a new set of corresponding points on the image curve as described in section 7 and compute the new image-model curve distance D_n.
- If $D_p - D_n < \varepsilon$, then STOP.
- Let $D_p = D_n$ and go to the second step above.

In this 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 Results and Discussion

A total of 15 model objects and seven input images were used to evaluate the object recognition system described in this paper. Three of those images and the system's corresponding output are shown in this section. The model objects are as following: *bottle, clip, fork, key, monkey wrench* (two model contours were used), *panda*, two *connector cases, screw-driver, scissors, spoon, vase, wire-cutter* and two regular *wrenches* (two model contours were used for each). Therefore a total of 18 model contours were used. Each model contour was acquired off-line by either manually reading and entering the coordinates of points on the contour or obtaining an image of the isolated model object, segmenting the image and recovering the contour. Each model contour was represented by 200 points. The segmentation of each model contour was also computed off-line. The exact same starting scale and final scale were used to compute the segmentation of each model contour. About 10-20 segments were extracted from each model contour.

Due to the light-box setup used, the images obtained had high contrast. As a result, thresholding followed by preprocessing was successful in properly segmenting each of the input images after which the bounding contours were recovered. Figure 9.1 shows three input images and the contours recovered from those. The left column shows the original input images, and the right column shows only the outermost contours recovered from those images after thresholding, processing and boundary detection (see section 2). Note that only the outermost contours were used by the system to arrive at recognition results even though the inner contours are visually significant to human viewers. This was done to demonstrate the recognition power of the system. Each image contour was represented by 300 points. The exact same starting scale and final scale were used to compute the segmentation of each image contour. About 20-40 segments were extracted from each image contour.

The input images depicted scenes of varying complexity. The scene depicted in the top row of figure 9.1 contains 4 objects and can be considered to be of medium complexity. The scenes depicted in the middle and bottom rows of figure 9.1 contain 6 and 8 objects respectively and can be considered difficult. The system was tested on each of the three inputs. Each of the objects in each input image was recognized correctly by the system which also determined the correct scale, location and pose of each object. *Note that none of the internal parameters of the program were modified from one run to the next: the exact same system produced the correct result for each input image.*

The system was implemented in *C* and ran on a *SiliconGraphics Crimson* workstation. Execution times of 3.9 seconds, 12.0 seconds, and 13.1 seconds were obtained for the top, middle and bottom scenes of figure 9.1 respectively. These execution times indicate that the system is very fast given the complexity of the tasks it must perform. The left column in figure 9.2 shows the recognition results reached by the system for the 3 input images. Note that in each sub-figure the model contours are shown using a thin line and the image contour is shown using a thick line. The system was very robust in each case despite the presence of noise and local deformations of shape due to segmentation errors, non-rigid material in some objects, and perspective projection. Note that the right column in figure 9.2 also shows the segmentation points discovered during recognition. In almost all cases, the system was able to determine exact locations of those points.

10 Conclusions

This paper presented a complete and practical system for object recognition with occlusion which is very robust with respect to noise and local deformations of shape (due to perspective projection, segmentation errors and material of a non-

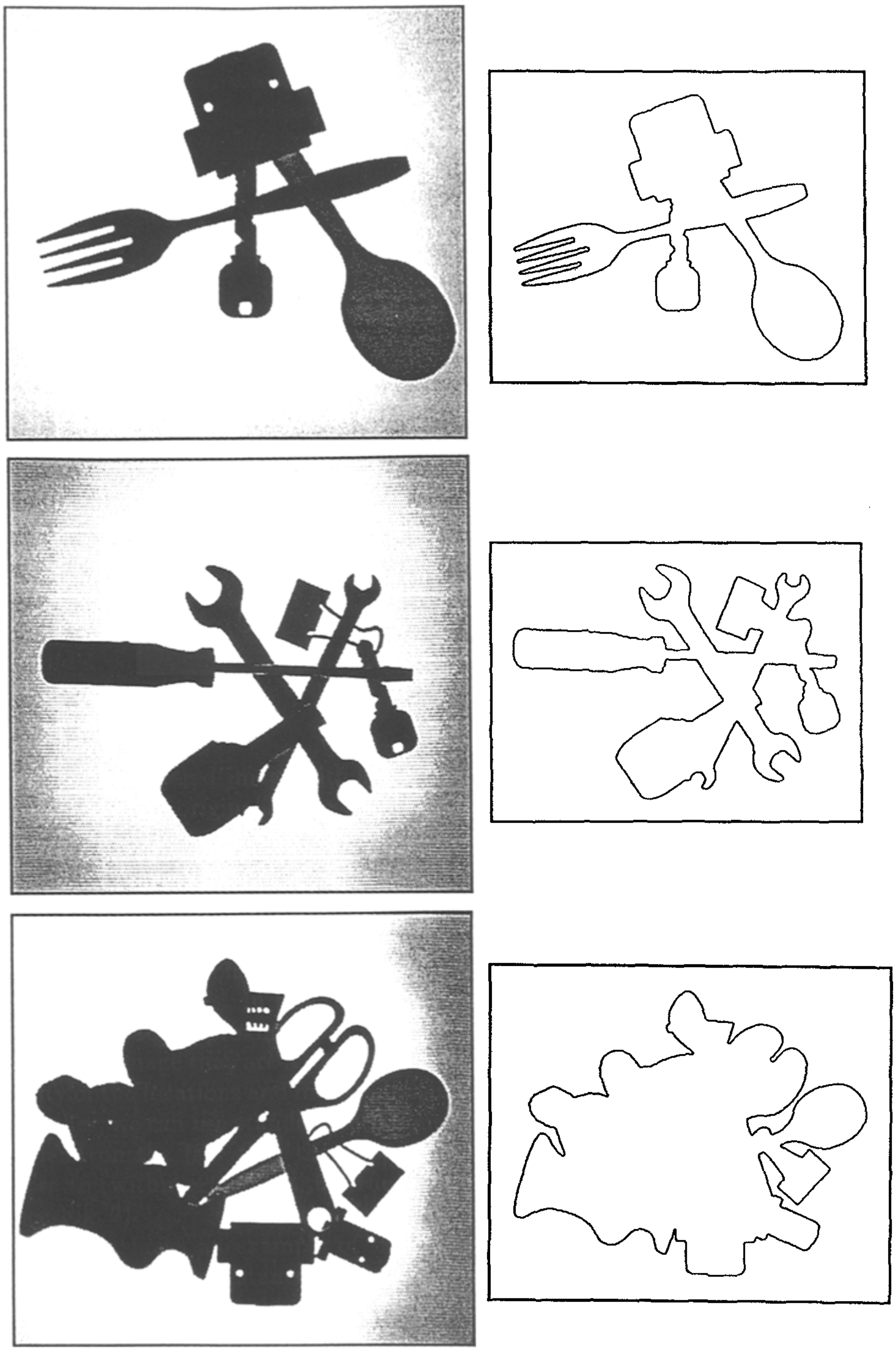

Figure 9.1. Input images and recovered contours

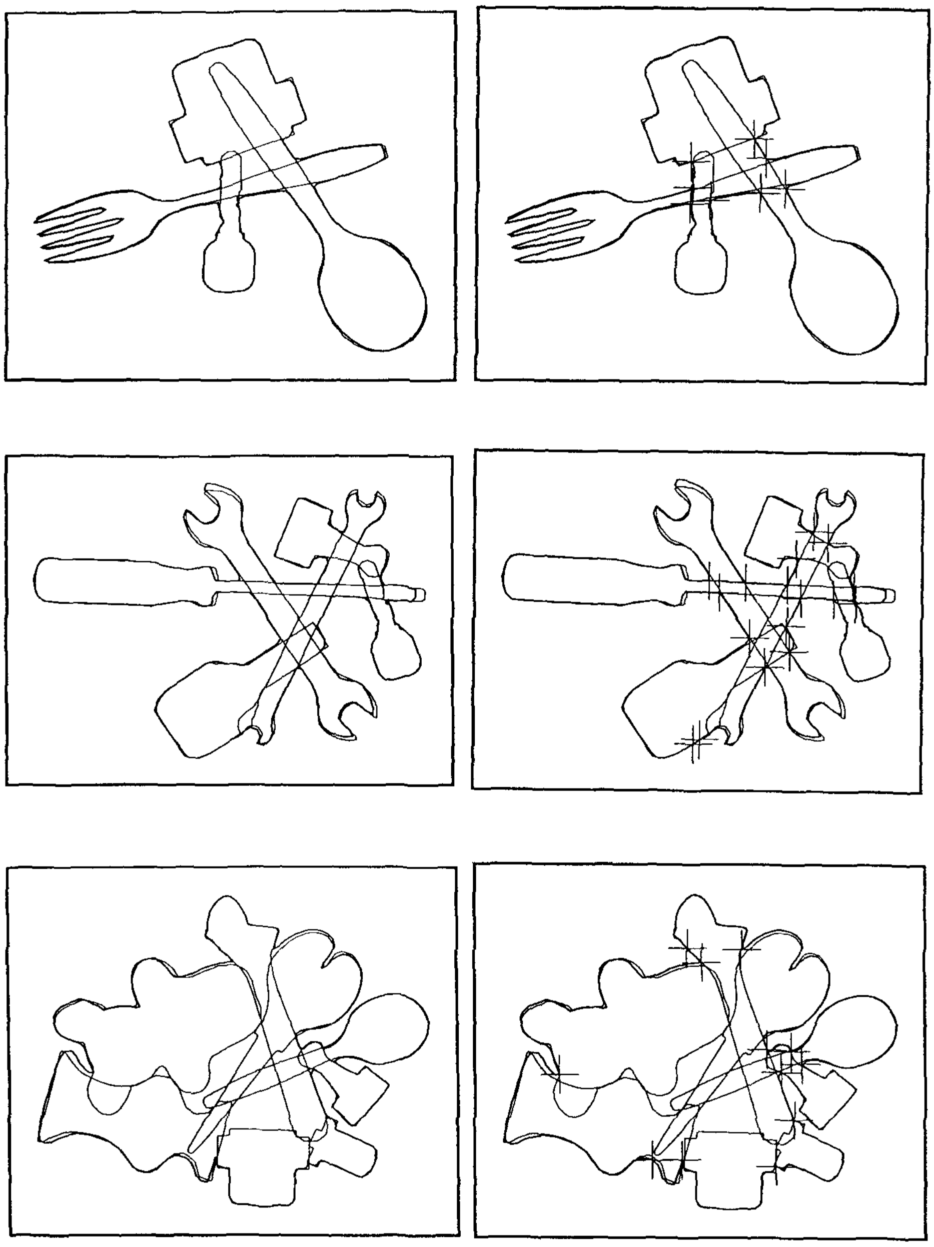

Figure 9.2. Recognition results for input scenes

rigid nature) as well as scale, position and orientation changes of the objects. The system was tested on a wide variety of 3-D objects with different shapes and surface properties. A light-box setup was used to obtain silhouette images which are segmented to obtain object boundaries. The *Curvature Scale Space* technique was then used to obtain a multi-scale segmentation of the image contour and the model contours using curvature zero-crossing points. This method made the system robust with respect to noise and local shape differences. A local matching algorithm applied *candidate generation, selection, merging, extension* and *grouping* to select the best matching models. Efficient transformation parameter optimization is used to map candidate models to the image space and directly measure the model-data quality of match. It is also used to compute the optimal pose for each selected model.

Acknowledgments

This work was supported by the NTT Basic Research Laboratories, Tokyo, the Laboratory for Computational Intelligence at the University of British Columbia, Vancouver, and the Vision, Speech, and Signal Processing Laboratory at the University of Surrey, England.

References

1. M. Kass, A. Witkin, and D. Terzopoulos. Snakes: active contour models. In *Proc International Conference on Computer Vision*, pages 259–268, 1987.
2. F. Mokhtarian. Fingerprint theorems for curvature and torsion zero-crossings. In *Proc IEEE Conference on Computer Vision and Pattern Recognition*, pages 269–275, San Diego, CA, 1989.
3. F. Mokhtarian. Silhouette-based isolated object recognition through curvature scale space. *IEEE Trans Pattern Analysis and Machine Intelligence*, 17(5), 1995.
4. F. Mokhtarian and A. K. Mackworth. Scale-based description and recognition of planar curves and two-dimensional shapes. *IEEE Trans Pattern Analysis and Machine Intelligence*, 8(1):34–43, 1986.
5. F. Mokhtarian and A. K. Mackworth. A theory of multi-scale, curvature-based shape representation for planar curves. *IEEE Trans Pattern Analysis and Machine Intelligence*, 14(8):789–805, 1992.
6. F. Mokhtarian and H. Murase. Silhouette-based object recognition through curvature scale space. In *Proc International Conference on Computer Vision*, pages 269–274, Berlin, 1993.
7. F. Mokhtarian and S. Naito. Scale properties of curvature and torsion zero-crossings. In *Proc Asian Conference on Computer Vision*, pages 303–308, Osaka, Japan, 1993.
8. C. A. Rothwell, D. A. Forsyth, A. Zisserman, and J. L. Mundy. Extracting projective structure from single perspective views of 3d point sets. In *Proc International Conference on Computer Vision*, Berlin, 1993.
9. F. Stein and G. Medioni. Structural indexing: Efficient 3-d object recognition. *IEEE Trans Pattern Analysis and Machine Intelligence*, 14:125–145, 1992.
10. A. P. Witkin. Scale space filtering. In *Proc International Joint Conference on Artificial Intelligence*, pages 1019–1023, Karlsruhe, Germany, 1983.