Curvature Scale Space Representation for Robust, Silhouette-Based Object Recognition with Occlusion

F. Mokhtarian[1]

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 (due to limited perspective projection, segmentation errors and soft surfaces) as well as scale, position and orientation changes of the objects. A wide variety of 3-D (not flat) objects with different shapes and surface properties have been used in order to test the system. No restrictive assumptions have been made about the shapes of admissible objects. An industrial application is envisaged where a fixed camera and a light-box are used to obtain images which are segmented in order to get the object boundaries. Every 3-D object can be modeled by a limited number of 2-D contours corresponding to the object's resting positions on the light-box. Note, therefore, that the system cannot be used for unconstrained 3-D object recognition from arbitrary viewpoints. The Curvature Scale Space (CSS) technique is then used to obtain a novel multi-scale segmentation of the image contour and the model contours using curvature zero-crossing points. Object indexing is used to narrow down the search-space. A local matching algorithm is utilized to select the best matching models. Efficient transformation parameter optimization is used to map candidate models to the image space and directly measures the model-data quality of match. It is also used to compute the optimal pose for each selected model.

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 occluded objects in the image reliably using their CSS representations [1]. The CSS representation is based on the scale space image concept introduced in [2] and popularized by [3]. It is an organization of curvature zero-crossing points on a contour at multiple scales. Previous publications by this author on the CSS representation [4,5] contain descriptions of how to compute CSS representations as well as theoretical investigations of the properties of those representations. Furthermore, a complete and robust isolated object recognition system based on the CSS representation was described in [6,7]. That system was designed specifically for recognition of isolated objects and, as a result, the system

1. Department of Electronic and Electrical Engineering, University of Surrey, Guildford, GU2 5XH, UK.

Scientia Iranica, Vol. 3, No. 4, © Sharif University of Technology, January 1997.

design is substantially different from the system presented here.

It is assumed that the recognition system developed here may be used for recognition of occluded three-dimensional objects. In particular, it is assumed that a number of 3-D objects are placed on a light-box in front of a camera (by a robot arm, for example) and that the task is to recognize each object. This particular task might be 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 object recognition tasks (such as in industrial settings).
- Every three-dimensional object, when resting on a flat surface and viewed by a fixed camera, has a limited number of classes of stable positions such that each class can be modeled using a two-dimensional contour. Note that the camera should not be very close to the objects. Furthermore, many 3-D objects can be effectively modeled using a finite number of characteristic 2-D views.
- Even with a light-box setup, recognition of 3-D objects can become challenging due to arbitrary shapes of those objects, noise and local deformations of shape which can be caused by perspective projection (when the camera is relatively close), segmentation errors and the soft surface of some objects.

The existing literature on shape representation and recognition is quite large. A detailed analysis or review is beyond the scope of this paper. A survey of some recent work can be found in [8]. Linear features such as points, lines and planes were used for 3-D object matching in [9]. Alignment was used for recognizing 3-D objects in 2-D images in [10]. An object recognition system was developed for bin-picking tasks in [11] which organized faces in an aspect graph. A data structure referred to as the feature sphere was developed in [12]. The Cresceptron framework was utilized in [13] for learning recognition of 3-D objects from 2-D images. Extended Gaussian images were used to determine object attitude in [14,15]. Registration of 3-D shapes was addressed in [16]. Sparse 3-D position and orientation measurements were utilized for object recognition in [17]. A Hopfield neural network was used for object recognition in [18]. Curved objects were recognized in [19] using a spherical representation. Interest features were used to create a hash-table in [20] which was utilized in conjunction with a generalized Hough transform technique. Shape polynomials were used in [21] to represent and compare shapes. The 2-D model-based object recognition system described in [22] employed transformation sampling. Symmetry-seeking models were made use of in [23] for 3-D object recognition. Overlapping parts were localized in [24] by searching the interpretation tree. An adaptive 3-D object recognition system was developed in [25] using multiple views. Illumination planning was employed in [26] for object recognition in structured environments.

In general, a shortcoming of some object representation techniques mentioned above 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. Another problem with some of these techniques is the limiting assumptions placed on the shapes of admissible objects. This paper presents a novel technique which addresses the shortcomings listed above by dealing effectively with free-form objects.

The first seven sections of this paper explain various aspects of the object recognition system developed. The order of presentation of these sections is the same as the order in which the tasks they explain are carried out by the system. The first section briefly explains how the processing of an image using a light-box system is accomplished. The next section reviews the CSS representation as a multi-scale

organization of the inherent features of a planar curve. The third section shows how multi-scale segmentation of a two-dimensional contour using curvature zero-crossing points may be accomplished. The fourth section describes a fast, local matching algorithm. The fifth section proposes a procedure for estimating the transformation parameters. The sixth section shows how the image-model curve distance can be computed and the last section describes a procedure for efficient optimization of the transformation parameters. Finally the results and an evaluation of the system, as well as the concluding remarks, are presented.

IMAGE PROCESSING

The use of a light-box setup makes the processing of the image reasonably straightforward. The same threshold value T was used to effectively segment all input images. Any pixel with a value less than T was classified as a 1-pixel, otherwise as a 0-pixel. A salt-and-pepper noise removal procedure was applied to the resulting binary image in order to remove isolated noise. This procedure sets any 1-pixel to zero which has more than 5 zero 8-neighbors and sets any 0-pixel to one which has more than 5 one 8-neighbors. It is applied to the image serially. This procedure is also referred to as median filtering. 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. This step corresponds to the application of mathematical morphology functions of dilation and erosion. The resulting binary image always has only one connected region of 1-pixels which corresponds to the objects. Next, boundary pixels belonging to the region of 1-pixels are detected: Any 1-pixel which has at least one zero 4-neighbor retains its value. Otherwise, it is set to zero. This procedure is applied in parallel. The final step is to recover the image coordinates of the boundary points. Here, this was done in a clockwise order. The image is scanned until the first boundary point is encountered. The eight neighbors of that point are searched in clockwise order starting with the top neighbor until a new neighbor is found. This procedure is repeated until the starting point is reached. The complete boundary curve is available.

THE CURVATURE SCALE SPACE REPRESENTATION

A CSS representation (or image) 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). It is, therefore, very suitable for representing the shapes of free-form objects. Curvature zero-crossings are points where the curvature function of a planar curve changes sign from positive to negative or from negative to positive. The CSS image of a planar curve represents this with ~ uniquely modulo scaling and a rigid motion [27]. To compute it, the curve Γ is first parametrized by the arc length parameter u [28]:

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

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 [29]:

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

where,

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

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

where $\otimes$ is the convolution operator and $g(u,\sigma)$ denotes a zero-mean Gaussian of width σ [30]. σ is also referred to as the scale parameter.

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 as an early form of active contours (snakes) [31], since they are similar in behavior 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, curvature needs to be computed 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,

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

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

$$Y_u(u,\sigma) = y(u)\otimes g_u(u,\sigma) ,$$

and:

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

The function defined implicitly by:

$$\kappa(u,\sigma) = 0 ,$$

is the CSS image of Γ. Note the following:

- As its name suggests, 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. Therefore, a curvature zero-crossing detected at scale σ at a contour point with arc-length value u is mapped to location (u,σ) in the CSS image.

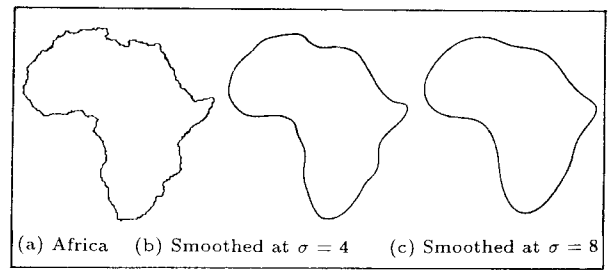


Figure 1. Africa during evolution.

- 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 usually used is 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.
- For all values of σ larger than 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 [32–34].

Figure 1 shows the coastline of Africa and two evolved versions of it. Figure 2 shows the CSS representation of Africa. For further examples of CSS images, see [5,35].

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 the next section). 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

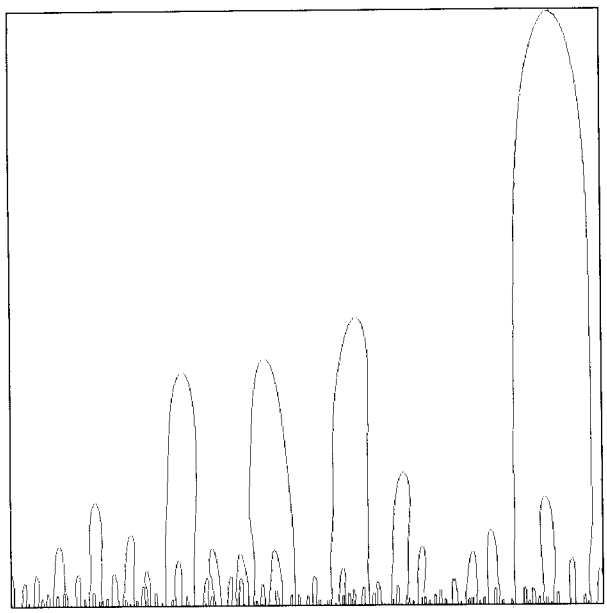


Figure 2. Curvature scale space representation of Africa.

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 CSS 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 for local matching.
- 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 (i.e., no segment can be found in the segment-list such that its left and right endpoints are close to the left and right endpoints of the new segment).

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.

As will be seen later, this multi-scale segmentation scheme is substantially more robust with respect to noise and local shape differences.

LOCAL MATCHING THROUGH CURVATURE SCALE SPACE

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

Rescaling

Model contours are rescaled so that they just fit in a unit square. They are further rescaled so that they reflect the relative sizes of model objects when viewed at the same distance. The aspect ratios of the model contours are not changed in the process. Model contour rescaling is carried out off-line. The image contour is also rescaled so that it just fits in a unit square (the image contour touches the square at 3 points or more). The aspect ratio of the contour is not changed in the process (note that only the outermost image contour is used for matching to demonstrate the recognition power of the system). 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 the previous section is then used to segment the model contours and the image contour.

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 [8,20,36] 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 is 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 corresponding to the possible match of each image contour segment and the model contour segments recovered from the index-table for that image contour segment.

Transformation parameter optimization is then applied (as described in later sections) 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 the sixth section) and is used as a measure of the goodness of fit between the two segments. A number of candidates with relatively low segment-dist values are then selected for further processing.

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. It must be emphasized that the test used here was not strict, since local matches can result in significantly different parameters even for compatible candidates.
- The corresponding segments of c_1 and c_2 must be neighboring or overlapping.
- The scale factor associated with the new merged candidate must be in the admissible space mentioned in the last subsection.
- 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. The algorithm described above will continue merging candidates until no two candidates can be found which satisfy the merging criteria.

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 on the image contour.
- The corresponding model contour segment is extended accordingly to determine its new endpoint.
- 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 at least one of the following conditions comes true:

- New candidate no longer has a low segment-dist value.
- New value of segment-dist rises sharply compared to previous value.
- 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.

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 the previous subsection 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 the process of recognition as will be described in the next section.

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 the last subsection to be compatible, the support of each candidate is increased by the length of the image contour segment associated with the other candidate). Therefore, the cost of each candidate is defined as following:

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

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 the previous subsection) with the chosen candidate.

- Find any image contour segments delimited by negative curvature minima (which are a subset of maxima of absolute value of curvature) 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 stops automatically and is independent of the number of objects in the input image.

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:

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

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 with four unknowns which is solved to obtain estimates for a, b, c and d:

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

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

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

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

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:

- As described in the previous sections, 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 the next section.

The following procedure is used to determine the image-model curve segment distance.

1. Let $k = 1$, η = number of vertices on model curve segment and $\delta = 0.0$.
2. Determine the closest point on the image curve segment (not necessarily a vertex) to vertex k of the model curve using the procedure described in Step 3.
3. Locate the closest vertex of the image curve segment to vertex k of the model curve. (All vertices of the image curve can be considered but to speed up the algorithm, only a small number in a neighborhood on the image curve segment into which vertex k maps, are considered.) Let that be point Q (Figure 3). Let points P and R be the image curve vertices which occur before and after Q on the image curve segment, respectively. Let point X be vertex k on the model curve segment. X can be any one of the points A, B, C or D in Figure 3. Cosines of the angles $\theta_1 = \angle XQP$ and $\theta_2 = \angle XQR$ can be computed using the definitions for the dot product of two vectors. Depending on the signs of the cosines, four cases are possible:
   - $\cos(\theta_1)$ and $\cos(\theta_2)$ are both negative. Point X is the same as point A in Figure 3. In this case, the closest image point is point Q.
   - $\cos(\theta_1)$ is positive and $\cos(\theta_2)$ is negative. Point X is the same as point B in Figure 3. In this case, the closest image point lies on line segment $P\overset{\frown}{Q}$. Since $\cos(\theta_1)$ is known, the location of that point as well as its distance to point B can be computed.
   - $\cos(\theta_1)$ is negative and $\cos(\theta_2)$ is positive. Point X is the same as point C in Figure 3. In this case, the closest image point lies on line segment QR. Since $\cos(\theta_2)$ is known, the location of that point and its distance to point C can be computed.
   - $\cos(\theta_1)$ and $\cos(\theta_2)$ are both positive. Point X is the same as point D in Figure 3. In this case, the closest image point lies on either PQ or QR. Compute the distances to both segments and choose the shorter distance and the corresponding point.
4. Let $\delta = \delta +$ distance computed in the previous step. Let $k = k+1$. If $k > \eta$ then return δ/η as the average point distance between the image and model curve segments and STOP. Otherwise, go to Step 2.

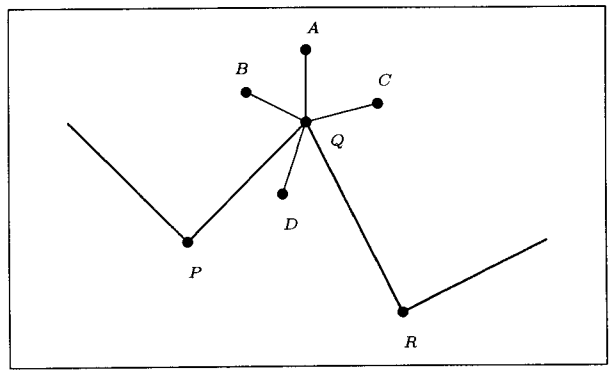


Figure 3. Closest point detection.

OPTIMIZING THE TRANSFORMATION PARAMETERS

The least-squares estimate of the transformation parameters computed in previous sections 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 using the following procedure:

- Let $D_p = \infty$.
- Compute the least-squares estimate of the parameters using the technique described previously 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 previously 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 is possible to compute the optimal parameters with less than 1% error using at most 10 iterations of the procedure described above.

RESULTS AND DISCUSSION

A total of fifteen model objects and seven input images were used to evaluate the object recognition system described in this paper. Six of those images and the system's corresponding output are shown in this section. The model objects are as follows: bottle, clip, fork, key, monkey wrench (two model contours were used), panda, two connector cases, screwdriver, scissors, spoon, vase, wire-cutter and two regular wrenches (two model contours were used for each). Therefore, a total of eighteen model contours were used. Figure 4 shows the model objects used to test the system. 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

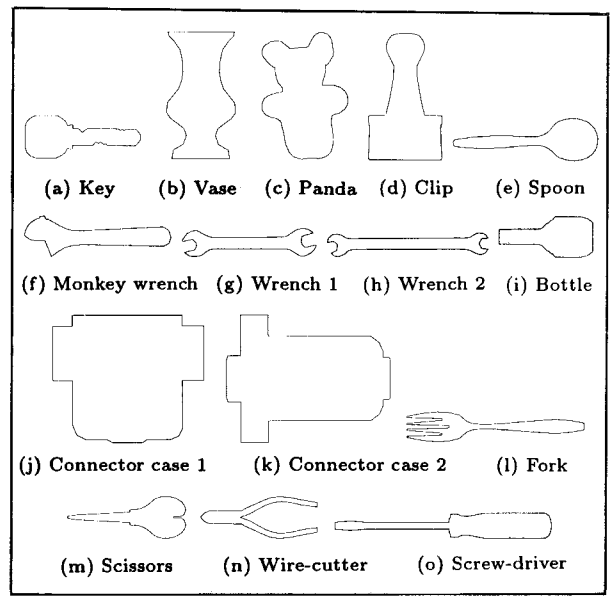


Figure 4. Model contours used to test the system.

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. Figures 5 through 10 show the initial processing of six input images. In each of those figures, sub-figure a shows the original input image and sub-figure b shows only the outermost contour recovered from that image after thresholding, processing and boundary detection (see the first section). Note that only the outermost contours were used by the system

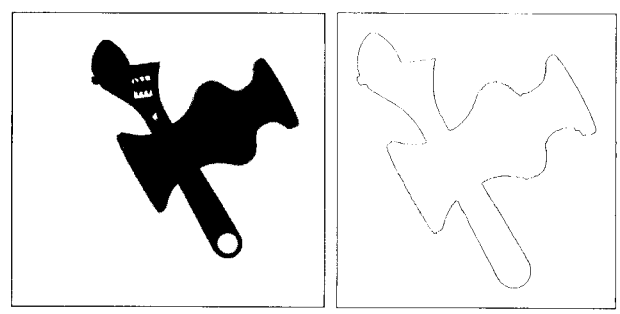
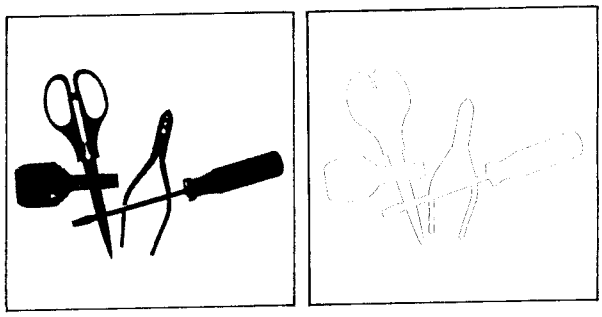
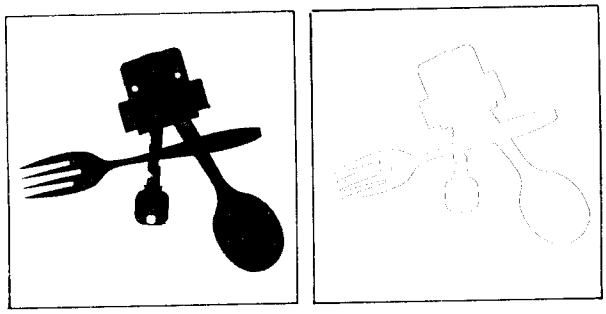


Figure 5. Initial processing of input Image 1.



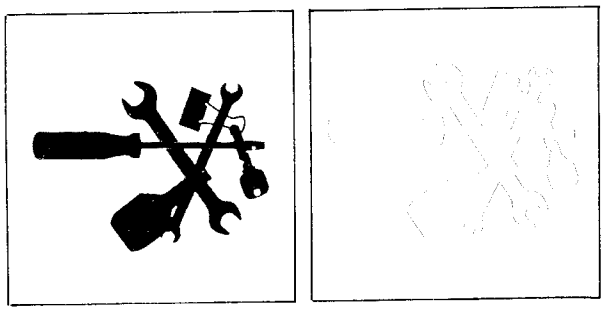
(a) Input Image 2 (b) Outermost contour from Image 2

Figure 6. Initial processing of input Image 2.



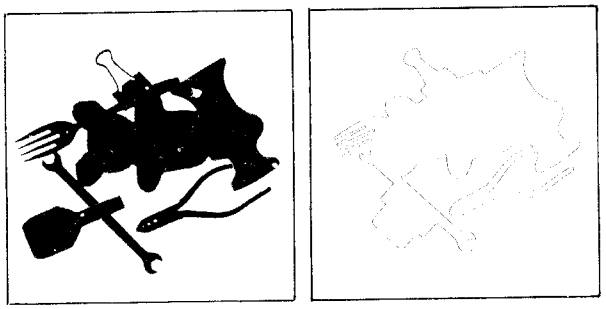
(a) Input Image 3 (b) Outermost contour from Image 3

Figure 7. Initial processing of input Image 3.



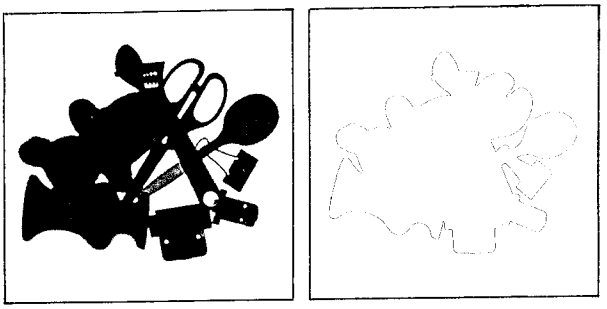
(a) Input Image 4 (b) Outermost contour from Image 4

Figure 8. Initial processing of input Image 4.



(a) Input Image 5 (b) Outermost contour from Image 5

Figure 9. Initial processing of input Image 5.

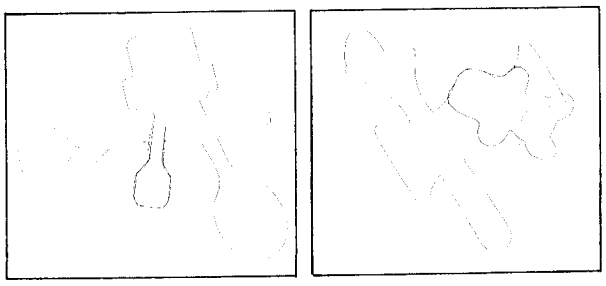


(a) Input Image 6 (b) Outermost contour from Image 6

Figure 10. Initial processing of input Image 6.

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.

In order to discover the correct scale, location and pose of each model matching the data, the system must consider all possible local matches (as indicated by the index-table) between all models and the data occurring at any scale in the admissible space. In doing so, quite frequently the system discovers locally plausible matches which are globally incorrect. Such situations are, in general, unavoidable and make the recognition task more challenging. Figure 11 illustrates this point. Figure 11a shows the bottle matched at an incorrect scale to the contour from Figure 7b. The correct model is the key. Figure 11b shows the panda



(a) Incorrectly matched bottle (b) Incorrectly matched panda

Figure 11. Locally plausible but incorrect matches.

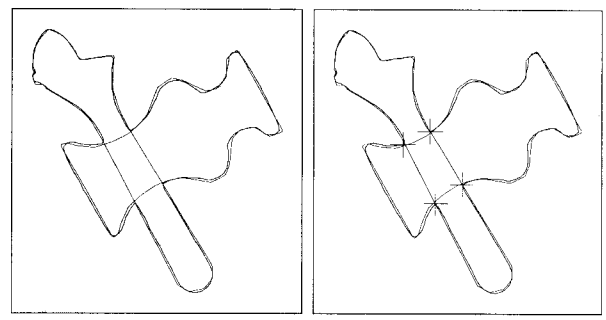
matched to the contour from Figure 5b. This example shows that even two objects (such as panda and vase), which appear to have very different shapes, can match well locally at the right scale and orientation. Note, however, that these were not the lowest-cost candidates chosen by the system. The lowest-cost candidates were in fact the correct ones, however, the two matches illustrated in Figure 11 ranked quite close to the lowest-cost candidates.

The input images depicted scenes of varying complexity. Image 5a depicts a simple scene with 2 objects. Images 6a and 7a each show 4 objects and can be considered to be of medium complexity. Images 8a, 9a and 10a which show 6, 7 and 8 objects, respectively, can be thought of as difficult images. The system was tested on each of the images 5a through 10a. 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 Silicon Graphics Crimson workstation. The running times obtained are given in Table 1 in seconds.

These running times indicate that the system is very fast given the complexity of the tasks it must perform. Figures 12a through 17a show the recognition results reached by the system for the six input images. Note that in each 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

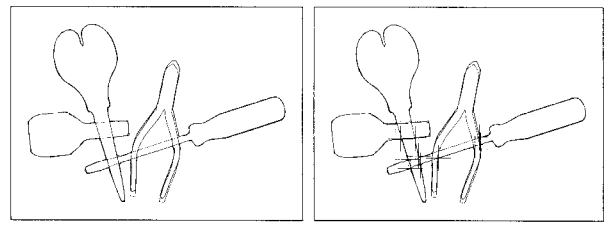
Table 1. System running times on input images.

Figure	Time (sec)
5b	3.5
6b	3.6
7b	3.9
8b	12.0
9b	10.1
10b	13.1



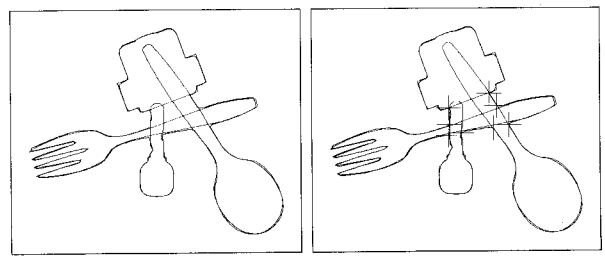
(a) Without segmentation (b) With segmentation

Figure 12. Matching result for Scene 1.



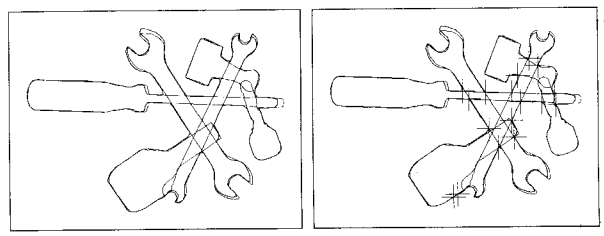
(a) Without segmentation (b) With segmentation

Figure 13. Matching result for Scene 2.



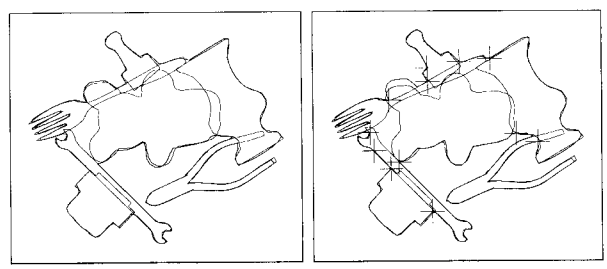
(a) Without segmentation (b) With segmentation

Figure 14. Matching result for Scene 3.



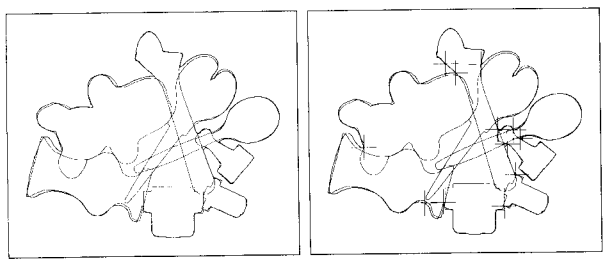
(a) Without segmentation (b) With segmentation

Figure 15. Matching result for Scene 4.



(a) Without segmentation (b) With segmentation

Figure 16. Matching result for Scene 5.



(a) Without segmentation (b) With segmentation

Figure 17. Matching result for Scene 6.

errors, the soft surface of some objects (such as the bear) and limited perspective projection (due to proximity of the camera). Figures 12b through 17b again show the same recognition results with segmentation points also displayed. In almost all cases, the system was able to determine the exact locations of segmentation points.

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 limited perspective projection, segmentation errors and the soft surface of some objects) 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 Basic Research Laboratories of the Nippon Telegraph and Telephone Corporation, Tokyo, Japan, the Laboratory for Computational Intelligence at the University of British Columbia, Vancouver, Canada and the Vision, Speech and Signal Processing Laboratory at the University of Surrey, England.

REFERENCES

1. Mokhtarian, F. and Mackworth, A.K. "Scale-based description and recognition of planar curves and two-dimensional shapes", *IEEE Trans Pattern Analysis and Machine Intelligence*, **8**(1), pp 34–43 (1986).
2. Stansfield, J. "Conclusions from the commodity expert project", Memo 601, MIT AI Lab, Cambridge, USA (1980).
3. Witkin, A.P. "Scale space filtering", *Proc. International Joint Conference on Artificial Intelligence*, Karlsruhe, Germany, pp 1019–1023 (1983).
4. Mokhtarian, F. and Mackworth, A.K. "Scale-based description and recognition of planar curves and two-dimensional shapes", *Computer Vision: Advances & Applications*, R. Kasturi and R.C. Jain, Eds., IEEE Computer Society Press, Los Alamitos, CA, USA, pp 154–163 (1991).
5. Mokhtarian, F. and Mackworth, A.K. "A theory of multi-scale, curvature-based shape representation for planar curves", *IEEE Trans Pattern Analysis and Machine Intelligence*, **14**(8), pp 789–805 (1992).
6. Mokhtarian, F. and Murase, H. "Silhouette-based object recognition through curvature scale space", *Proc. International Conference on Computer Vision*, Berlin, Germany, pp 269–274 (1993).
7. Mokhtarian, F. "Silhouette-based isolated object recognition through curvature scale space", *IEEE Trans Pattern Analysis and Machine Intelligence*, **17**(5) (1995).
8. Stein, F. and Medioni, G. "Structural indexing: Efficient 3-d object recognition", *IEEE Trans Pattern Analysis and Machine Intelligence*, **14**, pp 125–145 (1992).

9. Faugeras, O.D. and Hebert, M. "The representation, recognition, and locating of 3-d objects", *International Journal of Robotics Research*, **5**(3), pp 27–52 (1986).
10. Alter, T.D. and Grimson, W.E.L. "Fast and robust 3-d recognition by alignment", *Proc. International Conference on Computer Vision*, Berlin, Germany, pp 113–120 (1993).
11. Ikeuchi, K. "Precompiling a geometrical model into an interpretation for object recognition in bin-picking tasks", *Proc. DARPA Image Understanding Workshop*, Los Angeles, CA, USA, pp 321–339 (1987).
12. Chen, C.H. and Kak, A.C. "A robot vision system for recognizing 3-d objects in low-order polynomial time", *IEEE Trans Systems, Man and Cybernetics*, **19**(6), pp 1535–1563 (1989).
13. Weng, J.J., Ahuja, N. and Huang, T.S. "Learning recognition and segmentation of 3-d objects from 2-d images", *Proc. International Conference on Computer Vision*, Berlin, Germany, pp 121–128 (1993).
14. Little, J.J. "Determining object attitude from extended gaussian images", *Proc. International Joint Conference on Artificial Intelligence*, pp 960–963 (1985).
15. Horn, B.K.P. "Extended gaussian images", *Proc. of the IEEE*, **72**(12) (1984).
16. Besl, P.J. and McKay, N.D. "A method for registration of 3-d shapes", *IEEE Trans Pattern Analysis and Machine Intelligence*, **14**(2), pp 239–256 (1992).
17. Grimson, W.E.L. and Lozano-Perez, T. "Model-based recognition and localization from sparse range or tactile data", *International Journal of Robotics Research*, **3**(3), pp 3–35 (1984).
18. Nasrabadi, N.M., Li, W. and Choo, C.Y. "Object recognition by a hopfield neural network", *Proc. International Conference on Computer Vision*, pp 325–328 (1990).
19. Delingette, H., Hebert, M. and Ikeuchi, K. "A spherical representation for the recognition of curved objects", *Proc. International Conference on Computer Vision*, Berlin, Germany, pp 103–112 (1993).
20. Lamdan, Y. and Wolfson, H.J. "Geometric hashing: A general and efficient model-based recognition scheme", *Proc. International Conference on Computer Vision* (1988).
21. Taubin, G., Bolle, R.M. and Cooper, D.B. "Representing and comparing shapes using shape polynomials", *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp 510–516 (1989).
22. Cass, T.A. "A robust parallel implementation of 2d model-based recognition", *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp 879–884 (1988).
23. Terzopoulos, D., Witkin, A. and Kass, M. "Symmetry-seeking models and 3-d object recognition", *International Journal of Computer Vision*, **1**, pp 211–221 (1987).
24. Grimson, W. and Lozano-Perez, T. "Localizing overlapping parts by searching the interpretation tree", *IEEE Trans Pattern Analysis and Machine Intelligence*, **9** (1987).
25. Seibert, M. and Waxman, A.M. "Adaptive 3-d object recognition from multiple views", *IEEE Trans Pattern Analysis and Machine Intelligence*, **14**, pp 107–124 (1992).
26. Murase, H. and Nayar, S.K. "Illumination planning for object recognition in structured environments", *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, Seattle, WA, USA, pp 31–38 (1994).
27. Mokhtarian, F. "Fingerprint theorems for curvature and torsion zero-crossings", *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, San Diego, CA, USA, pp 269–275 (1989).
28. Goetz, A. *Introduction to Differential Geometry*, Addison-Wesley, Reading, MA, USA (1970).
29. Mackworth, A.K. and Mokhtarian, F. "Scale-based description of planar curves", *Proc. Canadian Society for Computational Studies of Intelligence*, London, Ontario, USA, pp 114–119 (1984).
30. Marr, D. and Hildreth, E.C. "Theory of edge detection", *Proc. Royal Society London B*, **207**, pp 187–217 (1980).
31. Kass, M., Witkin, A. and Terzopoulos, D. "Snakes: active contour models", *Proc. International Conference on Computer Vision*, pp 259–268 (1987).
32. Mokhtarian, F. and Naito, S. "Scale properties of curvature and torsion zero-crossings", *Proc. Asian Conference on Computer Vision*, Osaka, Japan, pp 303–308 (1993).

33. Mokhtarian, F. "Convergence properties of curvature scale space representations", *Proc. British Machine Vision Conference*, Birmingham, UK, pp 357–366 (1995).

34. Mokhtarian, F. "Zero-crossings of curvature and torsion in the limit", *Proc. Asian Conference on Computer Vision*, **III**, Singapore, pp 457–461 (1995).

35. Mackworth, A.K. and Mokhtarian, F. "The renormalized curvature scale space and the evolution properties of planar curves", *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, Ann Arbor, Michigan, USA, pp 318–326 (1988).

36. Rao, R.P.N. and Ballard, D.H. "Object indexing using an iconic sparse distributed memory" *Proc. International Conference on Computer Vision*, Cambridge, MA, USA (1995).