Silhouette-Based Isolated Object Recognition through Curvature Scale Space

Farzin Mokhtarian

Abstract—**A complete, fast 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 three-dimensional objects with different shapes and material 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* (CSS) *Representations* are computed for concave curves. The CSS representation is a multi-scale organization of the natural, invariant features of a curve (curvature zero-crossings or extrema) and useful for very reliable recognition of the correct model since it places no constraints on the shape of objects. A three-stage, coarse-to-fine matching algorithm prunes the search space in stage one by applying the CSS aspect ratio test. The maxima of contours in CSS representations of the surviving models are used for fast CSS matching in stage two. Finally, stage three verifies the best match and resolves any ambiguities by determining the distance between the image and model curves. Transformation parameter optimization is then used to find the best fit of the input object to the correct model.**

Index Terms— **Object recognition system, light-box setup, boundary contours, curvature scale space representation, maxima of curvature zero-crossing contours, coarse-to-fine matching strategy, transformation parameter optimization.**

I. Introduction

Object representation and recognition is one of the central problems in computer vision. Normally, a reliable, working vision system must be able to 1) effectively segment the image and 2) recognize objects in the image using their representations. This paper describes a complete, working vision system [8] which segments the image effectively using a light-box setup and recognizes isolated objects in the image reliably using their curvature scale space (CSS) representations [6], [7]. The CSS representation is based on the *scale space image* concept introduced in [10] and popularized by Witkin [14]. It is an organization of curvature zero-crossing points on a contour at multiple scales.

Note that an earlier CSS matching algorithm was implemented and tested in [6]. That algorithm was designed for both open and closed contour matching, made assumptions about the CSS image which were not always valid and was relatively slow. The CSS matching algorithm described in this paper in an improved, more efficient version of the earlier algorithm which has been designed specifically for closed contour matching.

It is assumed that the recognition system developed here may be used for recognition of isolated 3D 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. This particular task is believed to be interesting for the following reasons:

1) 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).
2) Every 3D 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 2D contour.
3) 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.
4) By considering only complete contour matching, a matching algorithm has been developed which is believed to be optimal for that particular task.

The existing literature on shape representation and recognition is quite large. Many methods are intended to be utilized on occluded scenes. There are also techniques designed for isolated object recognition. Examples are *Fourier descriptors* [9], [12], the *circular harmonic expansion* [1] and *moment invariants* [2], [3], [11]. One shortcoming of those techniques is that noise is not removed prior to the feature extraction process. As a result, only the first few low-frequency coefficients (or low-order moments) may be reliable for matching. This results in a coarse shape discrimination ability but would make it difficult to distinguish between objects with relatively small differences in shape. Furthermore, all techniques mentioned above compute global features of the input shapes and have no local support. Therefore local shape distortions cause global changes in the computed coefficients or moments. Another shortcoming is the difficulty of recovering the transformation parameters and point correspondences on the original contours. These would be useful to register those contours and verify that the correct match has been found or to choose between two or more close matches.

The following is the organization of the remaining sections of this paper. Section II explains the preprocessing carried out before matching starts. Section III describes a fast CSS matching algorithm. Section IV describes the verification steps taken after CSS matching. Section V gives an overall view of the implemented recognition system. Section VI presents the results and an evaluation of the system. Section VII contains the concluding remarks.

II. Preprocessing

This section describes the computations carried out by the system before matching begins. It consists of *image segmentation*, the *curvature scale space representation*, and *extracting maxima of curvature scale space contours*.

A. 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 color transparent material (tape-dispenser and screw-driver). A salt-and-pepper noise removal procedure was then applied to the thresholded image. Boundary pixels were then detected and marked. A contour following procedure was then used to recover the boundary curve (see Fig. 2).

B. The Curvature Scale Space Representation

A curvature scale space 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 [5]. To compute it, the curve Γ is first parametrized by the arc length parameter u:

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

Manuscript received June 7, 1993; revised April 28, 1994.

F. Mokhtarian is with the Department of Electronic and Electrical Engineering, University of Surrey, Guildford, Surrey, GU2 5XH, United Kingdom; e-mail: f.mokhtarian@ee.surrey.ac.uk.

IEEECS Log Number P95054.

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

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

where $*.$ is the convolution operator and $g(u, \sigma)$ denotes a Gaussian of width σ [4]. It can be shown that curvature κ on Γ_σ is given by [7]:

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

The curvature scale space image of Γ is defined as the solution to $\kappa(u, \sigma) = 0$. Note that the CSS representation is stored as a binary image and that *zero-crossing tracking* is used to speed up the computation of the CSS image. For examples of CSS images, see Figs. 3(c), 3(d), 4(c), and 4(d).

C. Extracting Maxima of Curvature Scale Space Contours

The CSS representation computed by the procedure described in the previous section is a binary image. 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 Figs. 3(c), 3(d), 4(c), and 4(d), a CSS contour is usually connected everywhere except possibly 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 greater computational cost.) So the actual maximum of a CSS contour usually falls in the *gap* at the top of that contour. In order to find such gaps, the CSS image is scanned to find pairs of zero-crossing points to qualify as the endpoints of gaps. To qualify, a pair of zero-crossing points much be close to each other and neither must have a zero-crossing neighbor at the next higher scale. When such a pair is found, the corresponding maximum is assumed to be the midpoint of the line segment joining the pair.

III. 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 scales 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. The reason for using the maxima as features is that they are the most significant points of zero-crossing 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 set is mere horizontal translation. 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 can be solved using a *best-first* matching strategy [13] 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.

The CSS matching algorithm is therefore as follows:

1) For each of the input CSS images, carry out the following: Extract the maxima of each CSS image. Record the coordinates of each maximum in a feature list as it is encountered. When the algorithm ends, this list will be sorted by the scale coordinate of the maxima. Normalize those coordinates so that the horizontal coordinate u varies in the range [0,1].
2) Create a number of nodes corresponding to the possible match of the highest-scale maximum of the image curve CSS and each maximum of the model CSS which has a σ-coordinate *close* (within 90%) to that of the highest model maximum. Initialize the *cost* of each node to zero.
3) For each node created in step 2, compute a CSS shift parameter α using the following formula:
$$u_m = u_i + \alpha$$
where u_i is the horizontal coordinate of the image curve CSS maximum and u_m is the horizontal coordinate of the model curve CSS maximum.
4) Create two lists for each node created in step 2. The first list will contain the image curve CSS maxima matched within that node at any point during program execution and the second list will contain the corresponding model curve CSS maxima. Initialize the first list of each node to contain the highest-scale image curve CSS maximum. Initialize the second list of each node to contain the corresponding model curve CSS maximum determined in step 2.
5) Expand each node created in step 2 one step using the procedure described in step 6.
6) To expand a node, select the highest-scale, image curve CSS maximum (which is *not* in its first list) and apply that node's CSS shift parameter computed in step 3 to map that maximum to the model CSS image. Locate the nearest model curve CSS maximum (which is *not* in the node's second list). The cost of match is defined as the straight line distance in the model CSS image between the two maxima. If there are no more image curve CSS maxima left, define cost of match as the height of the highest model curve CSS maximum *not* in the node's second list. Likewise, if there are no more model curve CSS maxima left, define cost of match as the height (after mapping) of the selected image curve CSS maximum. Add the match cost to the node cost. Update the two lists associated with the node.
7) Select the lowest-cost node. If there are no more model or image curve CSS maxima that remain unmatched within that node, then return that node as the lowest-cost node. Otherwise, go to step 6 and expand the lowest cost node.

IV. Verification

This section describes the verification steps carried out by the system after CSS matching. It consists of *solving for the transformation parameters, measuring image-model curve distances,* and *optimizing the transformation parameters.*

A. Solving for the Transformation Parameters

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

$$X = (x_j, y_j)$$

be a set of 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

$$x_j = a\xi_j + b\psi_j + c \qquad y_j = -b\xi_j + a\psi_j + d \tag{4.1}$$

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

$$\Omega = \sum (x'_j - x^c_j)^2 + (y'_j - y^c_j)^2$$

where (x^c_j, y^c_j) is the closest point on the model curve to transformed image curve point (x'_j, y'_j). Using (4.1) to eliminate x'_j and y'_j yields

$$\Omega = \sum (a\xi_j + b\psi_j + c - x^c_j)^2 + (-b\xi_j + a\psi_j + d - y^c_j)^2.$$

Let

$$P = (a, b, c, d)$$

be the vector defined by the transformation parameters. The solution of

$$\frac{\partial \Omega}{\partial P} = 0$$

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 can be solved to obtain estimates for a, b, c, and d.

B. 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:

1) Sometimes two or more model curves are close in shape. In such cases, it is useful to map the image curve to each of those model curves in order to determine which model curve is closest to the image curve. This is accomplished by measuring image-model curve distances.
2) The computation of the image-model curve distance is essential to transformation parameter optimization as described in Section IV.C.

The following procedure is used to determine image-model curve distance:

1) Let $k = 1$. Let η = the number of vertices on the image curve. Let $\delta = 0.0$.
2) Determine the closest point on the model curve (not necessarily a model curve vertex) to vertex k of the image curve. To speed up the algorithm, consider only a small neighborhood on the model curve into which vertex k maps.
3) Let $\delta = \delta +$ distance to closest point determined in step 2. Let $k = k + 1$. If $k > \eta$, then return δ as the total image-model curve distance and STOP. Otherwise, go to step 2

C. Optimizing the Transformation Parameters

The least-squares estimate of the transformation parameters computed in Section IV.A. 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:

1) $D_p = \infty$.
2) Compute the least-squares estimate of the parameters using the technique described in Section IV.A. and use it to map the image curve to the model curve.
3) Determine a new set of corresponding points on the model curve as described in Section IV.B. and compute the new image-model curve distance D_n.
4) $D_p - D_n < \varepsilon$, then STOP.
5) Let $D_p = D_n$ and go to step 2.

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.

V. A Silhouette-Based Object 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. Two hundred sample points are used on all model and image contours. The system can be divided into an *off-line* and an *on-line* component. The off-line part in completed first and is itself divided into two stages: *model acquisition* and *computing model representations*. Each one will now be described.

Model acquisition. Model contours can be obtained in two ways: manually entering the coordinates of points on the contours *or* obtaining an image of the model object, segmenting the image and recovering the contour. The former technique can result in less noise on the contour. Both techniques were used here to obtain the model contours. One contour must be obtained for each stable position of the model object (see Fig. 1). Each model contour is normalized so that it touches a square of size one on at least three sides.

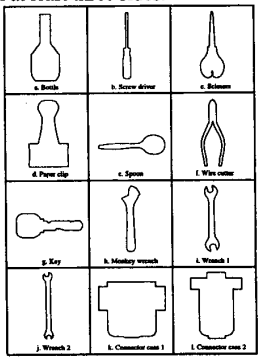

Fig. 1. Model contours used to test the system.

Computing model representations. If the model contour is concave, the following procedure is followed: The CSS representation of the model contour is computed off-line. The aspect ratio of the model contour CSS image is also computed. The maxima of the CSS representation

are then extracted. The final representation for the model contour is simply the CSS coordinates of each of those maxima. If the model contour is convex, that contour is smoothed until only four curvature extrema remain on the contour. The arc length coordinates of those fourextrema 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 can be divided into several stages as following:

1) The input image is first segmented to obtain a contour from the image. The contour is smoothed slightly to remove noise and normalized so that it touches a square of size one on at least three sides. 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 7 through 10. Otherwise, the input contour is considered to be concave and steps 2 through 6 are followed.
2) The CSS representation of the image curve is computed.
3) The maxima of the image curve CSS contours are extracted.
4) The aspect ratio of the image curve CSS is computed. Any model curve CSS image whose aspect ratio is close (within 10% but this figure can be higher) to the aspect ratio of the image curve CSS is accepted for step 5. Otherwise, it is rejected. This step can also be implemented using a hash table.
5) CSS matching (described in Section III) is applied to the surviving models.
6) The best 20% (but this figure can be higher) of the matches in step 5 are selected for verification. For each model curve which is selected, image curve transformation parameters are computed using their best CSS match and used to map the image curve to the model curve. The image-model curve distance is then computed. 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 used during the recognition process to find the best matching model.
7) The input convex contour is smoothed until only four curvature extrema remain.
8) 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 (within 10% but this figure can be higher) to this ratio is accepted for step 9. Otherwise, it is rejected. This step can also be implemented using a hash table.
9) 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.
10) 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 used during the recognition process to find the best matching model.

VI. Results and Discussion

The recognition system described in Section V was implemented in C and ran on a Silicon Graphics IRIS Crimson workstation. It was tested using a total of 22 model curves and 19 images. The following model contours were used to test the system: bottle, calculator, spray can lid, paper clip, fork, glue stick, key, monkey wrench (two sides), panda, two connector cases, screw driver, scissors, spoon, tape dispenser, vase, wire cutter and two wrenches (two sides each). All model contours were concave except the calculator, the glue stick and the spray can lid. Fig. 1 shows a number of the model contours used to evaluate system performance. Due to the light-box setup used, the images obtained had high contrast. As a result, thresholding was successful in properly segmenting each input image after which the bounding contours were recovered. Fig. 2 shows four input images and the contours recovered from those images.

Each one of the 19 input objects was recognized correctly by the system in less than one second (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, curve distance computation successfully resolved those ambiguities. The matching program is fast because it has been designed specifically for the task of matching closed curves using maxima of curvature scale space contours. These maxima (excluding the ones corresponding to very small CSS contours) are small in number (from 2 to 6 for the curves used as input to our system) but have a high multi-scale discriminative power. As a result, it is believed that they are the ideal feature points for the matching task considered here. It should also be pointed out that the CSS matching algorithm is a best-first algorithm which finds the best match of two CSS representations. It can be used in a serial fashion to find the best match of the image curve CSS to each of the surviving model curve CSS representations. It can also be run on parallel machines to find the best match to each model curve CSS representation on a separate machine. Finally, it can be embedded in another best-first algorithm which will match the image curve CSS to each model curve CSS one step at a time and always pursue only the best match. The latter technique would be the fastest serial implementation of the algorithm. Here, the first technique was used since the algorithm is already quite fast. Furthermore, the coarse-to-fine matching technique used makes efficient use of recognition power: models pass through *recognition filters* which become increasingly fine until the best-matching model is selected.

The thresholds used in the tests which act as filters between the various stages of the system (see steps 4, 6, and 8 in Section V) were drastically changed several times. 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 the following:

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

The following are 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). Fig. 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. The CSS images of the two curves are also shown. Fig. 4 shows the vase matched to the model vase. The local mismatch that can be observed is due to the fact that the

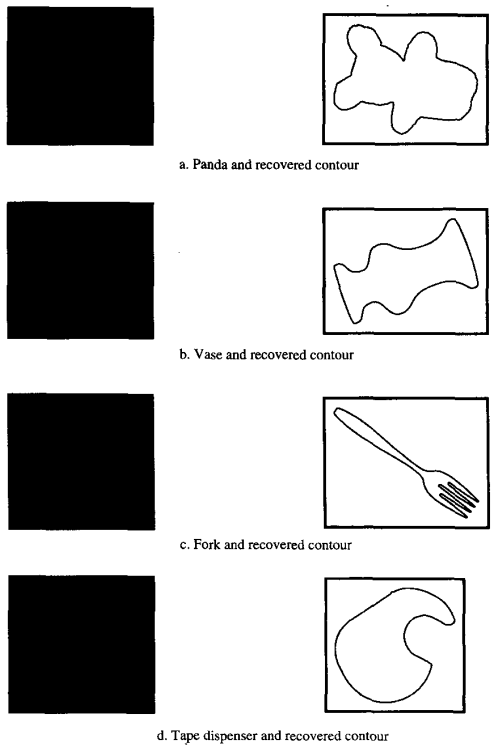

Fig. 2. Four input images and recovered contours.

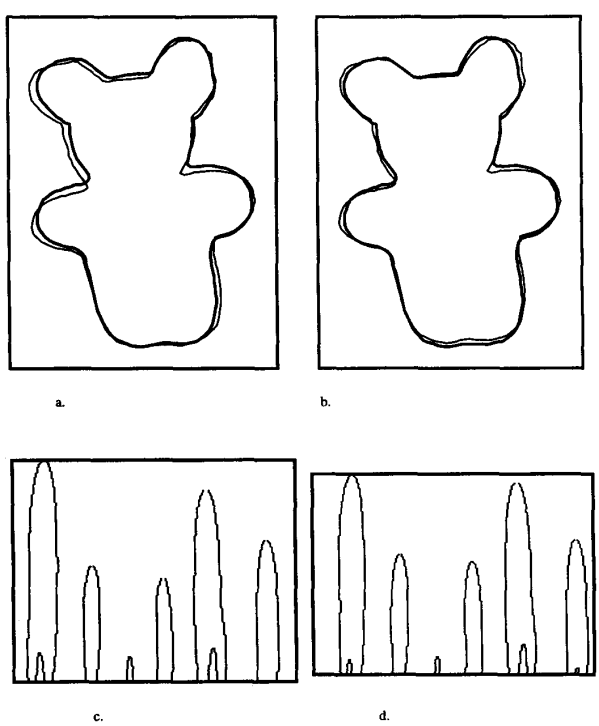

Fig. 3. (a) Panda matched to its model, (b) after transformation parameter optimization, (c) CSS image of image curve, (d) CSS image of model curve.

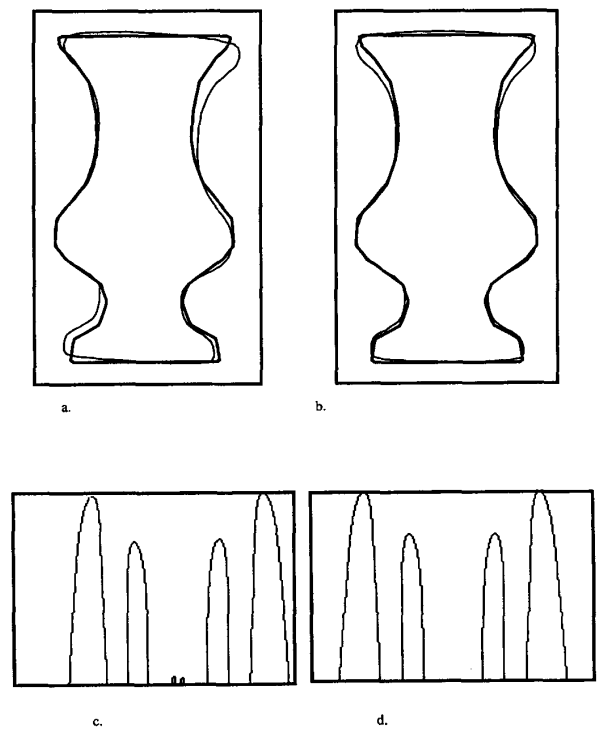

Fig. 4. (a) The vase matched to its model, (b) after transformation parameter optimization, (c) CSS image of image curve, (d)CSS image ofmodel curve.

model curve corresponds to an orthogonal projection of the vase whereas the image curve corresponds to its perspective projection. The CSS images of the two curves are also shown. The initial fit of the image curve to the model curve may not be very good due to noise or local shape deformations. Transformation parameter optimization will guarantee the best possible global fit.

It was discovered that, in general, if two contours had the same rough, coarse-level shape structure, then they would also have a close CSS match value. For example, this was true about the bottle and the clip contours. In such cases, image-model curve distance computation resolved the ambiguities. It was also discovered that a single model canbe used to represent a class of similar looking objects. For example, the model screwdriver was longer than the screwdriver used as input to the system.

VII. Conclusions

This paper described a complete, fast and practical isolated object recognition system which used a light-box setup to obtain silhouette images of objects. Those images were then segmented so that the physical boundaries of the objects could be recovered. Those boundaries were classified as either convex or concave. Convex curves were recognized using their four high-scale curvature extrema points. Curvature scale space representations were computed for concave curves. The CSS representation is a multi-scale organization of the invariant, natural features of a planar curve (curvature zero-crossings or extrema). A three-stage, coarse-to-fine matching algorithm was used to find the correct model for each concave image curve. The first stage applied the CSS aspect ratio test to prune the search space. The second stage used the maxima of CSS contours of the surviving models to quickly find the best-matching ones. The last stage verified the best match and resolved

any ambiguities by measuring the image-model curve distance in the model space. Finally, transformation parameter optimization was used to find the best fit of the input curve to the correct model. The system was tested on a variety of 3D objects with different shapes and surface and material 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.

ACKNOWLEDGMENTS

The author thanks S. Naito and H. Murase for helpful discussions.

This work was supported by the Basic Research Laboratories of the Nippon Telegraph and Telephone Corporation and the Laboratory for Computational Intelligence at the University of British Columbia.

REFERENCES

[1] Y. Hsu, H.H. Arsenault, and G. April, "Rotation-invariant digital pattern recognition using circular harmonic expansion," *Applied Optics*, vol. 21, no. 22, pp. 4012-4015, 1982.

[2] M.K. Hu, "Pattern recognition by moments invariants," *Proc. IRE*, vol. 49, 1961.

[3] A. Khotanzad and Y.H. Hong, "Invariant image recognition by Zernike moments," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 12, no. 5, pp. 489-497, 1990.

[4] D. Marr and E. Hildreth, "Theory of edge detection," *Proc. Royal Soc. London B*, vol. 207, pp. 187-217, 1980.

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

[6] 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*, vol. 8, no. 1, pp. 34-43, 1986.

[7] 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*, vol. 14, no. 8, pp. 789-805, 1992.

[8] F. Mokhtarian and H. Murase, "Silhouette-based object recognition through curvature scale space," *Proc. Int'l. Conf. Computer Vision*, pp. 269-274, Berlin, Germany, 1993.

[9] E. Persoon and K.S. Fu, "Shape discrimination using Fourier descriptors," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 7, pp. 170-179, 1977.

[10] J.L. Stansfield, "Conclusions from the commodity expert project," AI Memo No. 601, MIT AI Lab, Cambridge, Mass., 1980.

[11] C.H. Teh and R.T. Chin, "On image analysis by the methods of moments," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 10, no. 4, pp. 496-513, 1980.

[12] T.P. Wallace and O.R. Mitchell, "Analysis of three-dimensional movement using Fourier descriptors," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 2, pp. 583-588, 1980.

[13] P.H. Winston, *Artificial Intelligence*, Addison-Wesley, 1979.

[14] A.P. Witkin, "Scale space filtering," *Proc. IJCAI*, Karlsruhe, Germany, 1983.

Corrections

Corrections to "Pose Estimation by Fusing Noisy Data of Different Dimensions"

Y. Hel-Or and M. Werman

In the February issue of this transactions, in the above-mentioned correspondence (vol. 17, no. 2, pp. 195-201) the authors made a number of corrections that were not included in the final printed version. A complete corrected version of this paper is available at URL: http://www.cs.huji.ac.il/papers/IP/pose-estimation-pami.ps.gz.

Corrections to "Parts of Visual Form: Computational Aspects"

K. Siddiqi and B.B. Kimia

In the March issue of this transactions, in the above-mentioned paper (vol. 17, no. 3, pp. 239-251) there were some errors made during the final production process. They were:

1. Fig. 19 on page 245 was inverted. The outline figures should be on the bottom row.

2. Reference to citation [20] in the left column on page 246 should be deleted.

3. The sentence on co-circularity in the right column on page 246 should read, "The co-circularity [31] of only two pairs, $(T_l^1, T_l^2), (T_r^1, T_r^2)$, is computed, as only these are pairings of tangents lying on the same side of the part-line."

4. The fifth sentence in the left column on page 247 should have the following words added: "but pointing in opposite directions."

5. In the second sentence of the second paragraph on page 248, the word "outwards" should be inserted between the words "deforms" and "by."

6. Reference 18 should read:

[18] B.B. Kimia, A.R. Tannenbaum, and S.W. Zucker, "Entropy Scale-Space," *Visual Form: Analysis and Recognition*, C. Arcelli, ed. New York: Plenum Press, May 1991, pp. 333-344.