3D Object Recognition by Eigen-Scale-Space of Contours

Tim K. Lee[1,2] and Mark S. Drew[2]

[1] Cancer Control Research Program, BC Cancer Reserach Centre, 675 West 10th Avenue, Vancouver, BC, Canada, V5Z 1L3
[2] School of Computing Science, Simon Fraser University, 8888 University Drive, Burnaby, BC, Canada, V5A 1S6
tlee@bccrc.ca and mark@cs.sfu.ca

Abstract. People often recognize 3D objects by their boundary shape. Designing an algorithm for such a task is interesting and useful for retrieving objects from a shape database. In this paper, we present a fast 2-stage algorithm for recognizing 3D objects using a new feature space, built from curvature scale space images, as a shape representation that is scale, translation, rotation and reflection invariant. As well, the new shape representation removes the inherent ambiguity of the zero position of arc length for a scale space image. The 2-stage matching algorithm, conducted in the eigenspaces of the feature space, is analogous to the way people recognize an object: first identifying the type of object, and then determining the actual object. We test the new algorithm on a 3D database comprising 209 colour objects in 2926 different view poses, and achieve a 97% recognition rate for the object type and 95% for the object pose.

Keywords: object recognition; shape; 3D object; scale-space filtering; eigen-analysis

1 Introduction

Boundary contours of 3D objects contain rich information about the object. When such a contour is projected into our retinas as a planar 2D curve, we can often identify the object in spite of occlusions and accidental alignments. For example, Fig. 1 shows the 2D projections of the boundary contours of several objects from different viewing angles. Although no two of the closed curves are identical, we know all curves in the same column have similar shape and belong to the same type of object. In fact, we can name the objects as an axe, a baseball bat, a chair, a sailboat, a jet, and a bee. Designing a computer program to recognize 3D objects based on their projected boundary contours is interesting and useful for retrieving them from a database.

Building a shape recognition program requires two components: a shape representation and a matching algorithm. The curvature scale space (CSS) representation [1] has been shown to be a robust shape representation. Based on the scale space filtering technique applied to the curvature of a closed boundary curve,

the representation behaves well under perspective transformations of the curve. Furthermore, a small local change applied to the curve corresponds to a small local change in the representation, and the amount of change in the representation corresponds to the amount of change applied to the curve.

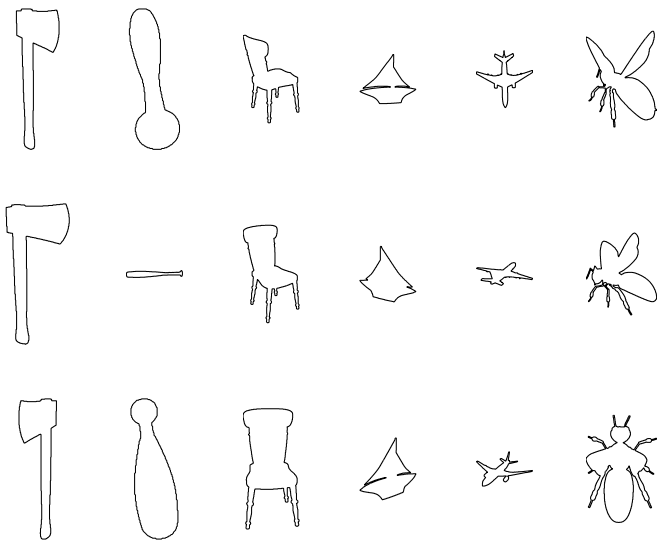

Fig. 1. Examples of the border contours for 3D objects from different viewing angles

More importantly, the representation supports retrieval of similar shape. Spurred partly by the success of the original CSS-derived shape retrieval algorithm [2], and because of the above properties, the CSS representation has been selected as the object contour-based shape descriptor for MPEG-7 [3]. In this paper, we propose an alternative fast matching algorithm for the CSS representation. The matching is carried out in the eigenspace of transformed CSS images. In spirit, then, this approach is inspired by Nayar et al.'s manifold projection scheme [4,5] for object and pose identification based upon appearance. In addition, matching images in eigenspace has been successfully applied to facial recognition [6], by finding an eigenface basis using a principal component analysis. Here, we are interested in developing eigenspaces that are expressive of CSS images, or rather eigenvectors of more compact expressions of these images. Objects can then be identified rapidly via a two-stage matching algorithm, which is an extension of our specialized shape matching algorithm [7]. The new algorithm is similar to the way in which many people recognize an object: first identifying the type of object, and then determining the actual object.

The paper is organized as follows. In §2 we briefly review the CSS representation. Section 3 presents a transformation applied to CSS images making them into a

compact and expressive representation that is *scale, translation, rotation and reflection invariant* and is suitable for a fast matching algorithm. Section 4 describes our 2-stage matching algorithm, and §5 reports experiment results on a 3D shape database. A short discussion and conclusions follow in §6.

2 Curvature Scale Space

The CSS representation relies on a binary 2D image, called the curvature scale space image, to represent the shape of a closed curve L_0 parameterized by t

$$L_0(t) = L_0(x(t), y(t)) \quad (1)$$

over multiple scales (see Fig. 2). The x dimension of the CSS image specifies the parameterized path length t of the curve and the y dimension specifies scales of the curve corresponding to the standard deviation σ of a Gaussian function $g(t, \sigma) = 1/\sigma\sqrt{2\pi}e^{-t^2/2\sigma^2}$. The binary CSS image is constructed by convolution of the closed curve $L_0(t)$ by a series of Gaussians $g(t, \sigma)$ with increasing σ, case given by

$$L(t, \sigma) = L_0(x(t), y(t)) \otimes g(t, \sigma) = (X(t, \sigma), Y(t, \sigma)) \quad (2)$$

where $\otimes$ denotes a convolution operation, $X(t, \sigma) = x(t) \otimes g(t, \sigma)$, and $Y(t, \sigma) = y(t) \otimes g(t, \sigma)$. The curvature functions $K(t, \sigma)$ of the smoothed curves $L(t, \sigma)$ are then calculated as

$$K(t,\sigma) = \frac{\frac{\partial X}{\partial t}\frac{\partial^2 Y}{\partial t^2} - \frac{\partial^2 X}{\partial t^2}\frac{\partial Y}{\partial t}}{\left[\left(\frac{\partial X}{\partial t}\right)^2 + \left(\frac{\partial Y}{\partial t}\right)^2\right]^{3/2}}. \quad (3)$$

For every zero-curvature point, i.e., $K(t, \sigma) = 0$ and $\partial K(t, \sigma)/\partial t \neq 0$, the corresponding location (t, σ) in the binary CSS image is set to 1. The markings of the zero-curvature points form a set of contours, whose appearance captures the shape of the closed curve $L_0(t)$. Fig. 2 shows an example of the smoothing process of a closed boundary curve and its corresponding CSS image.

3. Transformation of Scale Space Images

CSS images are bedevilled by an inherent ambiguity: the zero position of arc length is not determined for a new curve, compared to the model one. As well, reflections form a problem. To handle these rotation and mirror transformations of the boundary curve, and to improve the execution speed and matching performance, we propose applying the Phase Correlation method [8] along the abscissa (arc length) dimension of the CSS images. This transform aligns every curve's zero-position — the resulting curve

is a new kind of CSS image. We also need to compress the large amount of information in the CSS image in order to further speed up search. We form a new feature vector by summing separately over each of abscissa and ordinate (arc length and scale), and concatenating into a new feature vector, which we call the marginal-sum vector. This results in a feature vector not only invariant to rotations (starting position on the contour), but also invariant to reflections of the contour. Each of these steps is shown to result in a faster search and yet preserving the expressiveness of the original CSS formulation while simplifying the search procedure considerably. In addition, we propose conducting the matching in an eigenspace of reduced image vectors e.g. formed by the Singular Value Decomposition (SVD) method.

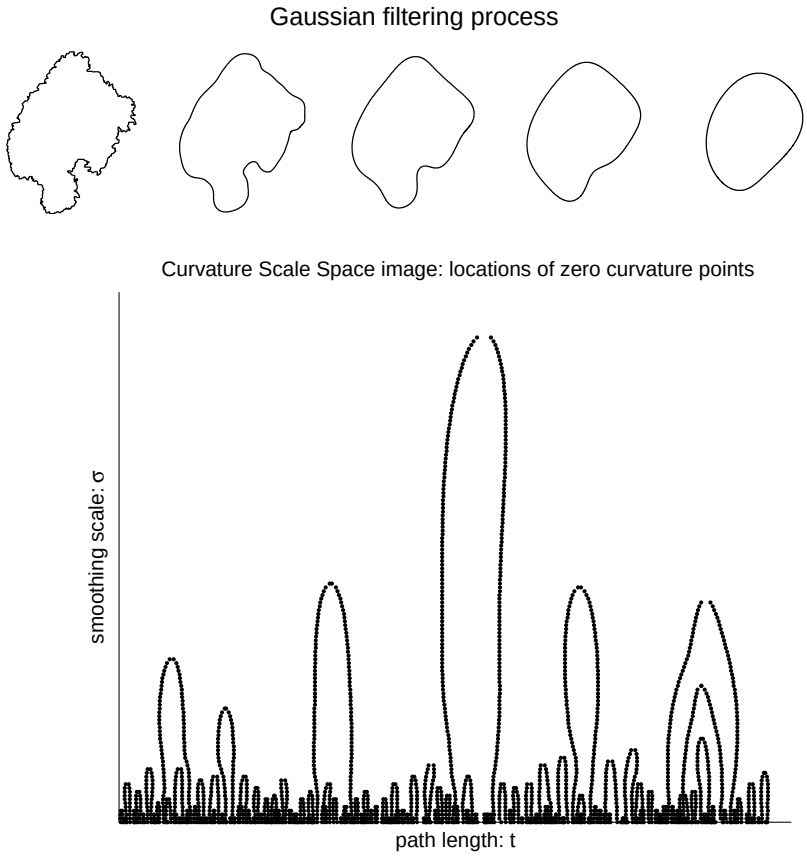

Fig. 2. (Top) Gaussian smoothing process of the leftmost closed curve. (Bottom) The corresponding curvature scale space image.

3.1 Eigenspace

Singular value decomposition is an efficient method for decorrelating vector information. For n column vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n \in \boldsymbol{R}^m$, we form an $m \times n$ data matrix $\boldsymbol{X}$ of mean-subtracted vectors

$$\boldsymbol{X} = [\boldsymbol{x}_1 - \bar{x}\ , \boldsymbol{x}_2 - \bar{x}\ , ..., \boldsymbol{x}_n - \bar{x}\ ]\ . \tag{4}$$

The SVD operation produces factors

$$\boldsymbol{USV} = \boldsymbol{X}\ , \tag{5}$$

with orthogonal $m \times m$ matrix $\boldsymbol{U}$ of eigenfeatures, $\boldsymbol{S}$ the $m \times m$ diagonal matrix of singular values, and $\boldsymbol{V}$ an $m \times n$ matrix of loadings. The column vectors of $\boldsymbol{U}$ form the basis for the eigenspace. In the new representation, vector $\boldsymbol{x}$ goes into a new coefficient m-vector $\boldsymbol{u}$ via

$$\boldsymbol{u} = \boldsymbol{U}^{\mathrm{T}} \boldsymbol{x}\ . \tag{6}$$

Since the eigenvectors are ordered by variance-accounted-for, we may often be able to reduce the dimensionality by truncating $\boldsymbol{u}$: if the number of bases used is reduced to k where $k < m$, the CSS eigenspace is truncated to $\boldsymbol{R}^k$**.** For any two vectors $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$, here we use a Euclidean metric as our distance measure.

3.2 Marginal-Sum Feature Vectors

The entire CSS image can be vectorized to form a column vector $\boldsymbol{x}_i$ of the input matrix $\boldsymbol{X}$ for the above SVD operation. However, such a formulation will create two problems. First, the resultant input matrix $\boldsymbol{X}$, which is an ensemble of all the vectors $\boldsymbol{x}_i$, will be very large and lead to long execution time for the SVD operation. Second, raw CSS images are too *noisy* for matching. The parameterized curves for the type of object that we intend to match may have slight alternations at arbitrary locations, and the CSS contour points will be unlikely to line up in their corresponding CSS images. Matching the CSS images pixel-by-pixel does not achieve the optimum result. Therefore we derive special column vectors $\boldsymbol{x}_i$, namely, marginal-sum feature vectors, from the CSS images. (We also had experimented on using the entire CSS; results are reported in [9]. The present method performed better than raw CSS image matching.)

Let $\boldsymbol{C}(i, j)$ denotes the pixel at the i^{th} row and j^{th} column of an r x c CSS image. The marginal-sum feature vector $\boldsymbol{x}$ is defined as a column vector composed of row-sum and column-sum vectors: $\boldsymbol{r}$ sums down the rows of the CSS image, and $\boldsymbol{c}$ sums across the columns.

$$\boldsymbol{x} = [\boldsymbol{r}\ \ \boldsymbol{c}]^{\mathrm{T}}\ , \tag{7}$$

$$\boldsymbol{r} = [\textstyle\sum_i \mathbf{C}(i, 1), \sum_i \mathbf{C}(i, 2), ... \sum_i \mathbf{C}(i, c)]^{\mathrm{T}}\ , \text{and} \tag{8}$$

$$\boldsymbol{c} = [\textstyle\sum_j \mathbf{C}(1, j), \sum_j \mathbf{C}(2, j), ... \sum_j \mathbf{C}(r, j)]^{\mathrm{T}}\ . \tag{9}$$

Vector $\boldsymbol{r}$ can be interpreted as the probabilities of zero curvature points, given a point t along the parameterized closed curve, while vector $\boldsymbol{c}$ denotes the probabilities of

zero curvature, given a smoothing level σ. Conceptually, we can view a feature vector $\boldsymbol{x}$ as the probability distribution function of the zero crossings of the CSS image.

3.3 Phase Correlation

Clearly, a rotation transformation on a closed boundary curve translates the initial point of the parameterization process, i.e., the CSS image is circularly shifted. On the other hand, a reflection transformation reverses the direction of the parameterization process, i.e., the CSS image is mirrored. These transformations pose a technical challenge to our algorithm; in particular, the vector $\boldsymbol{r}$ specified in eq. (8) will be affected, but the vector $\boldsymbol{c}$ specified in eq. (9) remains unchanged. Our solution to this problem is to carry out a phase correlation transform [8] on the vector $\boldsymbol{r}$, in the same way as Fourier phase normalization has been used to eliminate starting point dependency when matching contours using Fourier Descriptors [10,11]. This can be accomplished by converting the vector to the frequency domain, calculating the magnitude as a function of frequency, and transforming the results back to the spatial domain. The effect is translational alignment of the inverse-Fourier transformed functions. Note, however, that in carrying out this transform, we depart from a conventional CSS image, now going over to a phase-correlated version which is quite different from the original. But the phase correlation is carried out only in the abscissa, path-length direction, not in the ordinate, scale dimension. Mathematically, the phase-correlated vector $\widetilde{r}$ can be expressed as

$$\widetilde{r} = |F^{-1}(|F(\boldsymbol{r})|)|, \tag{10}$$

where F denotes a 1D Discrete Fourier Transform.

Therefore, we replace the marginal-sum feature vector in eq. (7) by

$$\boldsymbol{x} = [\, \widetilde{r} \;\; \boldsymbol{c}]^{\mathrm{T}}. \tag{11}$$

Notice that because of the nonlinearity of the absolute value operation, eq. (10) is *not* equivalent to forming a 2D CSS image which has been phase-correlated along the abscissa and then collapsed onto a row-sum. It is instead a much simpler operation. Our final representation is invariant to scale, translation, rotation and reflection transformation.

4. Two-stage Matching Algorithm

Our matching program is designed for a shape database with categorization information. Many shape databases collect more than one image for an object or a class of objects. These related images can be grouped into a category. For example, the 3D database from Michael Tarr's laboratory at the Department of Cognitive and Linguistic Sciences, Brown University [12] collects a set of 209 objects with 14 viewpoints. All views of an object form a category and we form a separate eigenspace

for each category. This turns out to be very effective with the following 2-stage recognition program: a high recognition rate is achieved even with a reduced subspace dimensionality.

4.1 Stage 1: Identify Image Category

Our recognition program is straightforward. In stage 1, we attempt to identify the category of a test object. This translates to the problem of finding the eigenspace that best describes the test object. To achieve this task, we project the feature vector of the test object (from Section 3) into each $\boldsymbol{R}^k$ category eigenspace in turn; the eigenspace giving the closest reconstruction of the test feature vector is defined as the best category. In other words, we are looking for the eigenspace that minimizes $\| \boldsymbol{x} - \hat{x} \|^2$, where $\boldsymbol{x}$ is the feature vector and $\hat{x} = \boldsymbol{U}\boldsymbol{U}^T\boldsymbol{x}$ is the reconstructed feature vector. By Parseval's Theorem, the distance metric can be calculated simply in terms of basis vector coefficients, not whole feature vectors, and so is fast.

4.2 Stage 2: Determine the Image in a Category

Once the best matched category is identified, we can determine the best matched object in the category by finding the object in the category with the minimum distance $\| \boldsymbol{u} - \hat{u} \|^2$, where $\boldsymbol{u}$ and $\hat{u}$ are the coordinates in $\boldsymbol{R}^k$ eigenspace for the test object and the object in the category, respectively.

5. Experiment Results

The algorithms presented in §3 and §4 have been implemented in Matlab and tested using the Tarr 3D database [12].

5.1 Test Data Set

The Tarr database consists of 209 objects such as 'axe', 'baseball bat', 'chair, 'sailboat', 'jet', 'bee', with each object captured from 14 different viewpoints: {0°, 30°, 60°, 90°, … 330°, bottom view, top view}, and are saved as 24-bit colour TIFF files. Fig. 3 shows the object 'axe' and 'bee'. Boundary contours are extracted using a simple contour-follower program [13], with segmentation results as in Fig. 4. Fig. 1 illustrates some other boundary contours for the database.

When we segment these objects from the background, we realize that some of images contain more than one objects and these objects are disjointed in one view but connected in other views (see Fig. 5.) In addition, highlights on some objects separate them into two objects after the segmentation (see Fig. 6.) In order to keep the

segmentation step simple and avoid human intervention, we segmented the images as-is and retained the border contours of only the longest-perimeter objects, without any subsequent processing. This has a tendency to work against the accuracy of our algorithm, so provides an extra level of rigour in testing.

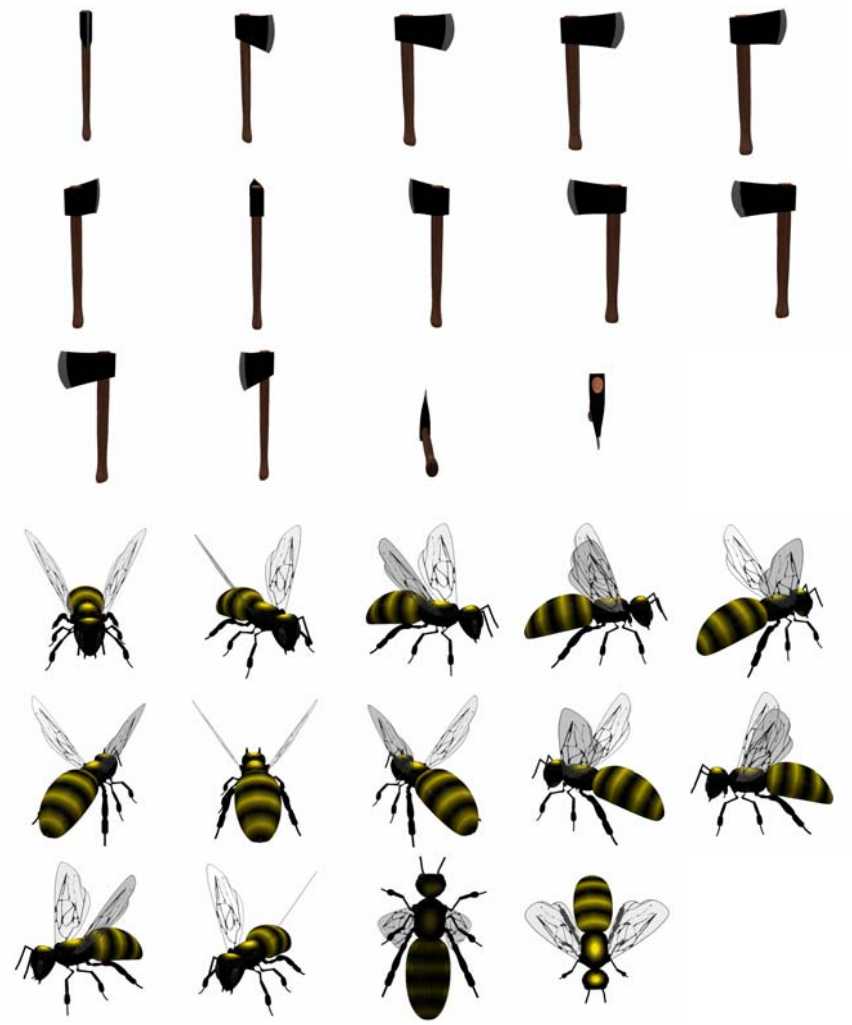

Figure 3. Two sample objects, axe and bee, in the Tarr 3D database. Each object is captured from 14 viewpoints at 0˚, 30˚, 60˚, …, 330˚, bottom view, and top view.

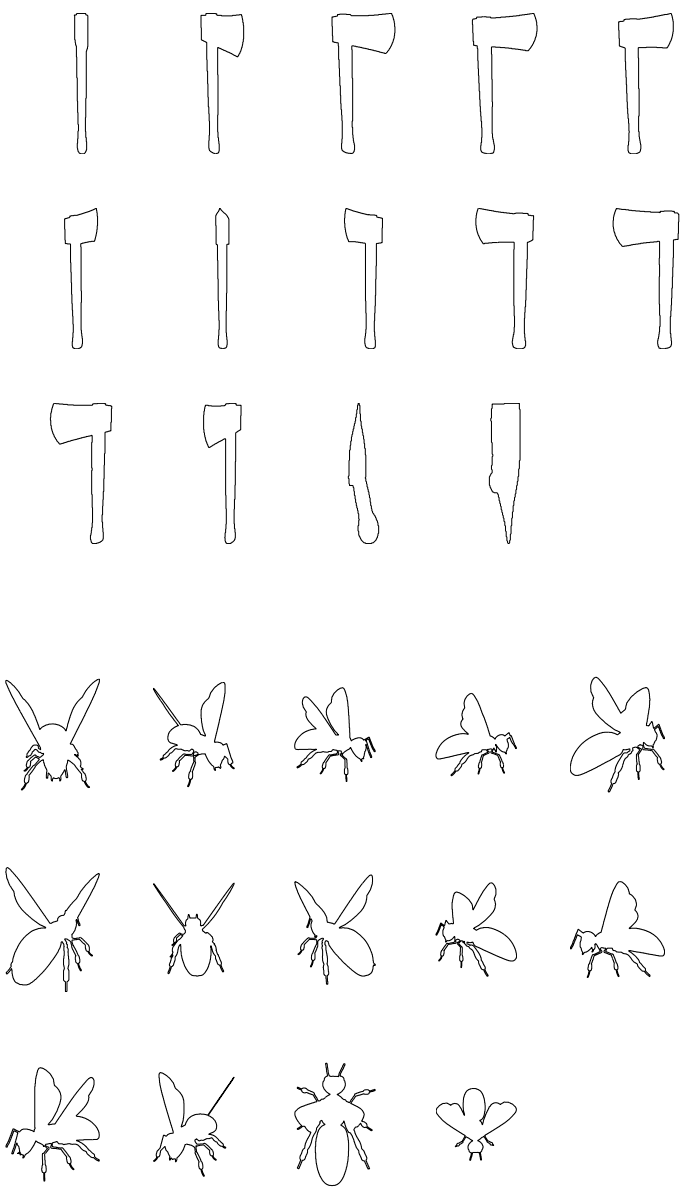

Fig. 4. The extracted boundary contours for Fig. 3. All contours are in fact closed curves, but slight gaps appearing in some of these figures are due to reproduction difficulties.

Fig. 5. A bottle and a glass are shown as separated objects in one view but as a connected object in another view. Our segmentation program retains only the border contour of the longest-perimeter connected object, e.g., just the bottle and not the glass for the leftmost image.

Fig. 6. The highlight on the second candle holder confused the simple segmentation program, which separated the holder from the candle. Thus, with our segmentation policy, only the candle holder was retained. The leftmost image of the candle holder and candle was segmented into one connected object.

5.2 Results

We computed the CSS image of every boundary curve by smoothing it with a series of Gaussian functions with starting σ value of 5, and incrementing by 1 until no zero curvatures remained on the smoothed curve. The small σ values (from 1 to 4), associated with fine texture irregularities of the curve, were removed from the CSS image to reduce the computation error introduced by a discrete boundary curve. As a result, the y-axis (the σ axis) of a CSS image had variable length, depending on the shape of the curve, but the x-axis had its length standardized to 200, the length of the parameterized curve. In order to standardize the size of all CSS images in a database, we padded the y-axis of all CSS images to 66, the maximum over all images.

The CSS images were used to construct the phase-correlated feature vector (eq. (11)). Grouping all phase-correlated marginal-sum vectors of the 14 views of an object into a matrix $\boldsymbol{X}$ (eq. (4)), we built 209 eigen-CSS eigenspaces, one for each object in the database, with the first $\boldsymbol{k}$ column vectors in SVD matrix $\boldsymbol{U}$, i.e., the eigenspace is in the subspace $\boldsymbol{R}^{\mathbf{k}}$.

We evaluated our recognition algorithm described in §4 by matching every boundary curve in a database against other curves in turn and determine whether our program can determine the test object's category (§4.1) and the specific object-pose (§4.2). Only the top best guess for the category and the pose itself were used to determine an average recognition rate, a stringent measure. For category identification

as described in §4.1, the average recognition rate was 92% for k = 5 (i.e. the eigenspace was reduced to R^5) and the rate improved to 97% for k = 13. When we calculated the recognition rate for specific object-pose (§4.2), it dropped slightly to 90% for k = 5 and 95% for k = 13. Fig. 7 shows the category and pose recognition rate for k = 1 to 14. We consider these results excellent for such a simple recognition algorithm.

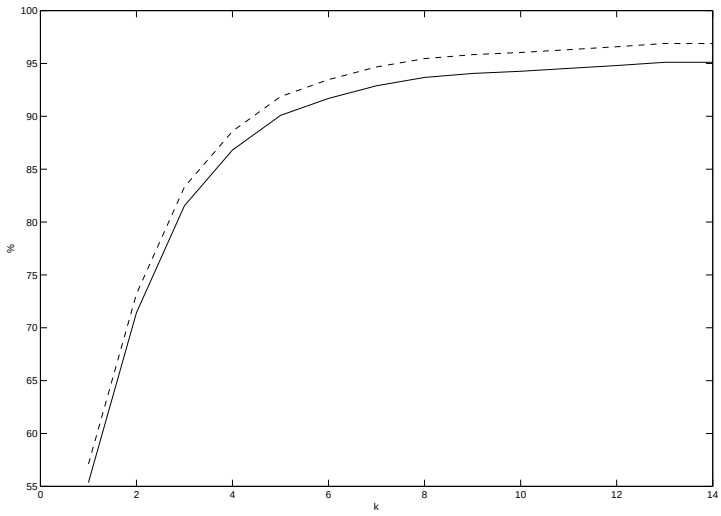

Fig. 7. Recognition rate for the Tarr 3D database: The dashed line shows the category recognition rate, while the solid line shows the pose recognition rate.

6 Conclusions

The advantages of our matching program are simplicity and execution efficiency. The matching algorithm is straightforward and simple, and is scale, translation, rotation, and reflection invariant. In addition, our shape representation compresses the CSS images and allows them to be processed rapidly: the feature vector for the Tarr 3D database consisted of only 266 elements. In conjunction with the reduced dimensionality in the eigenspace, the matching is extremely efficient.

The experiment results for the Tarr database also demonstrate the effectiveness of our algorithm. This database is challenging to work with. Beside the hurdles mentioned in §5.1 regarding multiple objects and highlights, the top and bottom views of an object often look markedly different from other views and, moreover, the front (0˚), back (180˚), 90˚ and 270˚ views can deviate substantially from other side views. Furthermore, many objects in the database have similar shapes (for example axes, flags, gavels, and pipes; batteries, brushes, carrots, chisels, crayons, flashlights, knifes, pens pencils, rulers, and screwdrivers; different types of chairs, etc).

Nevertheless, our matching program achieved a remarkable 97% recognition rate for categories and 95% for object-poses with only 13 eigen-bases, and not much less with only 5 bases. In sum, the method we present here is simple, fast and effective.

Acknowledgments. This work was supported in part by Discovery Grants from the Natural Sciences and Engineering Research Council of Canada (#288194 and # 611280).

References

1. Mokhtarian F., Mackworth A.: Scale-based description and recognition of planar curves and two-dimensional shapes, PAMI 8 (1986) 34–43.
2. Mokhtarian F.: Silhouette-based isolated object recognition through curvature scale space, PAMI 17 (1995) 539–544.
3. Mokhtarian F., Bober M.: Curvature Scale Space Representation: Theory, Applications, and MPEG-7 Standardization, Kluwer, 2003.
4. Murase H., Nayar S.: Learning object models from appearance, in: AAAI93, 1993, 836–843.
5. Murase H.,Nayar S.: Illumination planning for object recognition in structured environments, in: CVPR94 (1994) 31–38.
6. Turk M., Pentland A.: Face recognition using eigenfaces, in: Computer Vision and Patt. Recog.: CVPR91 (1991) 586–591.
7. Drew M.S., Lee T.K., and Rova A.: Shape retrieval with eigen-css search, Image and Vision Computing, submitted Feb. 24, 2005.
8. Kuglin C., Hines D: The phase correlation image alignment method, in: Int'l Conf. on Cybernetics and Society (1975) 163–165.
9. Drew M.S., Lee T.K., and Rova A.: Shape retrieval with eigen-css search, School of Computing Science, Simon Fraser University, Burnaby, BC, Canada Technical Report TR 2005-07, 2005.
10. Arbter K., Snyder W., Burkhardt H., Hirzinger G.: Application of affine invariant Fourier descriptors to recognition of 3-D objects, IEEE Trans. Patt. Anal. and Mach. Intell. 12 (1990) 640–647.
11. Kunttu I., Lepist L., Rauhamaa J., Visa A.: Recognition of shapes by editing shock graphs, in: Int. Conf. on Computer Vision: ICCV2001 (2001) 755–762.
12. Tarr M.J.: The object databank. Available from http://alpha.cog.brown.edu:8200/stimuli/objects/objectdatabank.zip.
13. Sonka M., Hlavac V., and Boyle R: Image Processing, Analysis and Machine Vision, Chapman & Hall Computing (1993), p. 129.