Shape Retrieval with Eigen-CSS Search

Mark S. Drew, [a,∗,1] Tim K. Lee, [b,a] Andrew Rova [a]

[a]*School of Computing Science, Simon Fraser University, Vancouver, Canada V5A 1S6*

[b]*Cancer Control Research, BC Cancer Research Centre, Vancouver, Canada, V5Z 1L3*

Abstract

Shape retrieval programs are comprised of two components: shape representation, and matching algorithm. Building the representation on scale space filtering and the curvature function of a closed boundary curve, curvature scale space (CSS) has been demonstrated to be a robust 2D shape representation. The adoption of the CSS image as the default in the MPEG-7 standard, using a matching algorithm utilizing maxima of the CSS image contours, makes this feature of interest perforce. In this paper, we propose a framework in two stages for a novel approach to both representing and matching the CSS feature. Our contribution consists of three steps, each of which effects a profound speedup on CSS image matching. Each step is a well-known technique in other domains, but the proposed concatenation of steps leads to a novel approach to this subject which captures shape information more efficiently and decreases distracting noise. Firstly, using experience derived from medical imaging, we define a set of marginal-sum features summarizing the CSS image. Secondly, the standard algorithm using CSS maxima involves a complicated and time-consuming search, since the zero of arc length is not known in any new contour. Here, we obviate this search via a phase normalization transform in the spatial dimension of the reduced marginal-CSS feature. Remarkably, this step also makes the method rotation- and reflection-invariant. Finally, the resulting feature space is amenable to dimension reduction via subspace projection methods, with a dramatic speedup in time, and as well orders of magnitude reduction in space. The first stage of the resultant program, using a general-purpose eigenspace, has class-categorization accuracy compatible with the original contour maxima program. In a second stage, we generate specialized eigenspaces for each shape category, with little extra runtime complexity because search can still be carried out in reduced dimensionality. In a leave-one-out categorization using the MPEG-7 contour database, a classification success rate of 94.1% over 1400 objects in 70 classes is achieved with very fast matching, and 98.6% in the top-2 classes. A leave-all-in test achieves 99.8% correct categorization. The method is rotation invariant, and is simple, fast, and effective.

Key words: Shape; 2D contour; scale-space; matching; retrieval; curvature; CSS; eigen-analysis.

1 Introduction

An object's closed boundary curve contains rich information about the object. Examining the contour, we can recognize the object shape and often identify the type of object. For example, Fig. 1 shows the boundary curves of several objects. Although no two 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 birds, camels, forks, hammers, and elephants. This paper aims at recognizing the object category, not retrieving a specific object. The method presented produces impressive results, nearly as good as the best method available, but is very simple and consequently fast.

Building a shape retrieval 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. 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 shape retrieval algorithm based on the CSS representation. The matching is carried out in the eigenspace of reduced, transformed CSS image feature vectors. 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. 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 as belonging to the most likely subspace for categories of objects.

We need to compact the large amount of information in the CSS image in order to further speed up search. Motivated by the medical imaging tomographic technique of reducing dimensionality by summing, 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 we call a marginal-sum vector. This removal of information can actually benefit classification, similarly to how compression of color histograms can remove noise and improve image retrival [6].

* Corresponding author.
Email addresses: `mark@cs.sfu.ca` (Mark S. Drew,), `tlee@bccancer.bc.ca` (Tim K. Lee,), `arova@cs.sfu.ca` (Andrew Rova).
[1] Mailing address: School of Computing Science,
Simon Fraser University
8888 University Drive,
Vancouver, B.C. CANADA V5A1S6

Tel: 1-778-782-4277
Fax: 1-778-782-3045

As well, 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. Moreover, 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 apply the Phase Correlation method [7] to marginal sums of CSS image columns, i.e., along the abscissa (arc length) dimension of the original CSS images. This transform aligns every curve's zero-position — the resulting curve is a new kind of CSS image feature. To our knowledge, this has not been done before for CSS image feature vectors.

This results in a feature vector which is 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 that does not give up the expressiveness of the original CSS formulation while simplifying the search procedure considerably.

The paper is organized as follows. In §2 we briefly review the CSS representation and the standard shape retrieval algorithm. Section 3 describes our new matching algorithm, and §4 reports experiment results for object categorization. The advantages of the algorithm are discussed and conclusions are drawn in §5.

2 Synopsis of CSS Matching by Contour Maxima

2.1 CSS Representation

The CSS representation [1] relies on a binary 2D image, the curvature scale space image, to represent the shape of a closed curve L_0 parameterized by t,

$$L_0(t) = L_0(x(t), y(t)) \tag{1}$$

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 smoothed by a Gaussian with standard deviation σ: $g(t,\sigma) = 1/(\sigma\sqrt{2\pi})\, e^{-t^2/2\sigma^2}$. The binary CSS image is constructed by first generating the convolution of the closed curve $L_0(t)$ by a series of Gaussians with increasing σ,

$$L(t,\sigma) \;=\; L_0(x(t), y(t)) \otimes g(t,\sigma) \;=\; (X(t,\sigma), Y(t,\sigma)) \tag{2}$$

where $\otimes$ denotes convolution, $X(t,\sigma) = x(t) \otimes g(t,\sigma)$, 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

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

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 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.

2.2 Matching by CSS Contour Maxima

The canonical CSS-based shape retrieval algorithm [2,3,8] compares a test CSS image with a set of CSS model images in an image database, returning a subset of similar model images. The similarity measure is defined as the total distance between the corresponding contour *maxima* in the two CSS images. A perfect match has a zero difference-measure.

In order to find the minimum cost of the match between a test image and a model, the algorithm must consider all possible ways of aligning the *high-scale* contour maxima from both CSS images, and compute the associated cost. For every possible candidate pair of contour maxima, there are two ways to align them: either shifting the test CSS circularly in the horizontal direction or shifting the model CSS. Because of the asymmetric treatment of the test and model CSS images, both alignment methods must be attempted and their associated costs estimated separately.

Unfortunately, the above procedure fails to detect the mirror-image of the input image, even if it is in the database. Therefore, the algorithm has to repeat once again, by comparing the mirrored CSS test image with all the models. Finally, all costs of the match must be considered to calculate the closeness in appearance for all models.

Pseudocode for the algorithm is as follows:

```
Standard algorithm:

Loop for all models
    Cost for model is the minimum of (
        matchCSS(image, model),
        matchCSS(model, image),
        matchCSS(mirror(image), model),
        matchCSS(model, mirror(image)))
End loop
Rank models based on their costs

Function matchCSS(css1, css2)
    Loop for all contour maximum pairs of css1 and css2
        Align css1 and css2 by shifting css1 horizontally
        Determine cost of the match
    End loop
    Return(minimum cost among all pairs)
End function
```

2.3 Class Matching Evaluation Method

Good performance was reported when the above algorithm was evaluated for image databases [3]. As an example, Fig. 3 shows a database of 131 fish used for evaluation. These are divided into 17 classes (0 to 16). An image database is composed of a set of boundary curves, pre-assigned into classes based on their shapes. One of the boundary curves is used to match shape-to-shape against the other curves stored in database, and the first 15 best matches are returned. The number of returned matches *belonging to the same class* as that of the input is divided by the number of curves in that class to obtain the performance measure for the input curve. To derive the performance measure over the class, the above step is repeated for all curves in the same class. And finally, the performance measure for all classes is determined.

3 Matching by Eigen-CSS

In this section, we describe Stage 1 and Stage 2 of our shape retrieval algorithm, which again rely on CSS images to represent object shapes. However, instead of examining CSS image contour maxima, we propose conducting the matching in an eigenspace of reduced image vectors. For our application, motivated by medical imaging tomographic applications we convert CSS images into more compact and descriptive feature vectors we call marginal-sum vectors, which are phase-aligned and then further projected into an eigenspace (for Stage 1) or class eigenspaces (for Stage 2) and hence truncated to just a few coefficients.

In Stage 1 of our algorithm, we use a general-purpose eigenspace pertaining to the whole database. In Stage 2, we create specialized eigenspaces each pertaining to a single object class.

3.1 Marginal-Sum Feature Vectors

The entire 2-D binary CSS image could be vectorized to form a column vector $\mathbf{x_i}$ of an input matrix $\mathbf{X}$ for a Singular Value Decomposition (SVD). However, such a formulation would have two problems. First, $\mathbf{X}$ would be very large and lead to long execution time for the SVD operation. Second, raw CSS images may be too *noisy* for matching. The parameterized curves for the type of object that we intend to retrieve have small alternations at arbitrary locations, and the CSS contour points will be unlikely to line up in their corresponding CSS images. So matching the CSS images pixel-by-pixel may not be able to achieve the optimum result. Therefore we posit removing some of this noise by deriving alternative vectors $\mathbf{x_i}$, that we call *marginal-sum feature vectors,* from the CSS images. [2]

To motivate the following, consider medical imaging tomographic applications. There, suppose we wish to consider 2D skin pathologies. Doctors often consider very simple measures of skin structures

[2] We have also experimented with the use of the entire CSS; results are reported in [9]. The present method performed better than did raw CSS image matching by Euclidean distance.

based on the lengths of a few diameters across the lesion. A more complicated 3D case might involve 2D images formed from various 3D directions. For our skin lesion situation, the analogue would be many 2D x-rays perpendicular to the lesion, or in other words *sums* over material from a particular ray direction. The same basic idea applies here: we can glean a good deal of information about structures by simply considering sets of sums across them. Here, we apply this concept by treating each 2D CSS image as a structure we wish to examine. Instead of the raw data, we consider the set of row-sums and set of column-sums, concatenated, as a novel feature vector. The utility of this choice is borne out by the experiments below, and in fact the new vector does indeed seem to nullify to some extent the effects of noise, resulting in better shape categorization.

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

$$\mathbf{x} = [\mathbf{r}\ \mathbf{c}]^T : \tag{4}$$

$$\mathbf{r} = [\sum_i \mathbf{C}(i,1), \sum_i \mathbf{C}(i,2), ... \sum_i \mathbf{C}(i,c)]^T, \quad \text{and} \tag{5}$$

$$\mathbf{c} = [\sum_j \mathbf{C}(1,j), \sum_j \mathbf{C}(2,j), ... \sum_j \mathbf{C}(r,j)]^T \tag{6}$$

Vector $\mathbf{r}$ can be interpreted as the probabilities of zero-curvature points, given a point t along the parameterized closed curve, while vector $\mathbf{c}$ denotes the probability of zero curvature, given a smoothing level σ. Conceptually, we can imagine the feature vector as formed by collapsing the CSS image in the y and x directions separately, precisely as in a tomographic x-ray process.

3.2 *Phase Normalization*

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. Figs. 4(a,b) show a 180° rotation and a vertical mirroring transformation of a boundary curve, along with their corresponding CSS images. These transformations pose a technical challenge to our algorithm; in particular, the vector $\mathbf{r}$ specified in eq. (5) will be affected, but the vector $\mathbf{c}$ specified in eq. (6) remains unchanged. Our solution to this problem is to carry out a phase normalization transform [7] on the vector $\mathbf{r}$, 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 $\mathbf{r}$ 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. In carrying out this transform we depart from a conventional CSS image, now going over to a phase-normalized version which is quite different from the original. The phase normalization is carried out only in the abscissa, path-length direction, not in the ordinate, scale dimension. Since we carry out the phase normalization only on $\mathbf{r}$, i.e, part of the marginal-sum feature vector, and not on the full CSS image, phase normalization is in fact very simple.

Hence, a phase-normalized vector $\tilde{\mathbf{r}}$ can be expressed as

$$\tilde{\mathbf{r}} = |F^{-1}(|F(\mathbf{r})|)| \tag{7}$$

where F denotes a 1D Discrete Fourier Transform. Fig. 4(c) shows the vectors $\mathbf{r}$ for the corresponding CSS images. In Fig. 4(d) , the phase-normalized vectors $\tilde{\mathbf{r}}$'s are plotted — these three vectors should have the same values, since as shown below in § 3.2.1, in fact eq. (7) produces a feature vector *invariant to rotations and reflections*.

Overall, therefore, we replace the marginal-sum feature vector in eq. (4) by

$$\mathbf{x} = [\tilde{\mathbf{r}}\ \mathbf{c}]^T \tag{8}$$

Notice that because of the nonlinearity of the absolute value operation, eq. (7) is *not* equivalent to forming a 2D CSS image which has been phase-normalized along the abscissa and then collapsed onto a row-sum. It is instead a much simpler operation.

3.2.1 Mirror reflections

A "vertical mirroring" of a contour, reflecting around the middle vertical column as in Fig. 4(a), is effected by reversing the order of the abscissa coordinates, and hence by reversing vector $\mathbf{r}$. Suppose N is the number of columns in the CSS image. Then the discrete Fourier transform $\hat{F}_u$ of the row-sum $\hat{\mathbf{r}}$ of the flipped contour $\hat{C}$ is found via

$$\begin{aligned}
\hat{r}_n &= \textstyle\sum_i \hat{C}(i,n) \;=\; r_{(N+1-n)};\\
F_u &= \textstyle\sum_{n=1}^{N} r_n\, exp(-2\pi i(u-1)(n-1)/N), \quad u = 1..N;\\
\hat{F}_u &= \textstyle\sum_{n=1}^{N} r_{(N+1-n)}\, exp(-2\pi i(u-1)(n-1)/N)\\
&= \textstyle\sum_{n=1}^{N} r_n\, exp(+2\pi i(u-1)(n-1)/N)\, exp(2\pi i(u-1)/N)\\
&= conj(F_u)\, exp(2\pi i(u-1)/N)
\end{aligned} \tag{9}$$

Thus we have that $|F_u|$ = $|\hat{F}_u|$ and the two transforms differ only by a phase.

Therefore the feature $\tilde{\mathbf{r}}$ in eq. (7) is invariant to vertical mirroring. As well, clearly the sums across columns, $\mathbf{c}$, are also unchanged, so the complete feature vector $\mathbf{x}$ is invariant.

For "horizontal mirroring", wherein the contour is inverted from top to bottom, the CSS image is unchanged except that it is reversed left-right and shifted. Thus again we have that the Fourier transform of the column-sums down the rows differ only by a phase, and $\tilde{\mathbf{r}}$ is invariant; as well, clearly the row-sums $\mathbf{c}$ across the columns are unchanged.

3.3 Algorithm Structure

We constructed two different versions of the algorithm. Stage 1 investigates applying the CSS image transformation and dimensional reduction scheme using a general eigenvector basis. This could be used in a situation in which categorization of classes in a target database were not available. We find below, in § 4, that class categorization accuracy of this approach is comparable to, or better than, the standard algorithm, which is based on matching maxima.

In a second stage, however, if class metadata is available, we can further develop an eigenbasis separately for each class. Projecting onto each class provides a powerful mechanism for classifying new, test CSS images. The price for Stage 2 processing is a mild increase in both space and time complexity, as we shall see below. And the categorization results are impressive, comparable to the best algorithm available but very simple and fast.

3.3.1 Stage 1: General Eigenspace

Our Stage 1 shape retrieval algorithm is divided into two modules: an eigenspace construction module, and a matching module. Pseudocode for eigenspace construction is as follows:

```
Stage 1:

Obtain phase-normalized marginal-sum vector for each CSS
Form input matrix for SVD
Perform SVD, reducing to $K$ dimensions
Map each training set CSS into $\mathcal{R}^K$ eigenspace
```

Matching an image is done by computing the corresponding phase-normalized marginal-sum feature vector $\mathbf{x} = [\tilde{\mathbf{r}}\ \mathbf{c}]^T$, and mapping it into the $\mathcal{R}^K$ eigenspace. Similarity between a test feature vector and the models is measured by their Euclidean distances in the eigenspace. Then success with respect to class categorization is measured by the class matching evaluation method developed in § 2.3.

3.3.2 Stage 2: Specialized Eigenspaces

If a categorization of the boundary-curve database is available, a further refinement can be carried out. For example, in the MPEG-7 shape database there are 70 classes with 20 objects each. We can form a separate eigenspace for each of these 70 categories, and then match a transformed CSS feature vector with each subspace separately. This turns out to be quite effective. As well, the subspace dimensionality required is found to be very low to still produce effective categorization.

In sum, the algorithm, including eigenspace construction and matching modules, is then modified to the following:

```
Stage 2:

Obtain phase-normalized marginal-sum vector for each CSS
Form input matrix for SVD for each object category
Perform multiple SVDs
Map test CSS into each dim-$k$ eigenspace $\mathcal{R}^k$
The eigenspace that gives the closest reconstruction of the test CSS
   feature vector is the best category
```

By Parseval's Theorem, the matching can in fact be carried out in the reduced, eigen-coefficient domain. We wish to characterize a good fit for a test feature vector $\mathbf{x}$ with the approximation $\hat{\mathbf{x}}$ by considering the Euclidean distance $\|\mathbf{x} - \hat{\mathbf{x}}\|^2$. Now suppose the k-vector set of coefficients in a subspace is $\boldsymbol{\epsilon}$: $\hat{\mathbf{x}} = \sum_{i=1}^{k} \epsilon_i \mathbf{x}_i$, where $\mathbf{x}_i$ is the ith eigenvector. Then our distance measure, for determining how well a test feature vector $\mathbf{x}$ is expressed in the eigen-subspace, is given by

$$\begin{aligned} D^2 &= \|\mathbf{x} - \hat{\mathbf{x}}\|^2 = (\mathbf{x}^T - \textstyle\sum_{j=1}^{k} \mathbf{x}_j^T \epsilon_j)(\mathbf{x} - \sum_{i=1}^{k} \mathbf{x}_i \epsilon_i) \\ &= \mathbf{x}^T\mathbf{x} - \|\boldsymbol{\epsilon}\|^2 \end{aligned} \tag{10}$$

assuming the eigenvectors are orthonormal so that $\epsilon_j \equiv \mathbf{x}_j^T \mathbf{x}$. Since the first term above is fixed, for a particular test image, we minimize distance D by simply maximizing the sum of squares of subspace coefficients $\boldsymbol{\epsilon}$.

The meaning of this is that the CSS image feature vector is best represented by the subspace in which its coordinates are best mapped. The idea is similar to finding the best 2D plane of several available to best characterize a 3D vector: the best 2D subspace is that for which orthogonal projection of the 3-vector produces the largest-magnitude projections.

While production of the object category eigenspaces does take time offline, the runtime complexity for matching is not much increased by using a Stage 2 algorithm: For the Stage 1 algorithm, using a general basis for all shapes, suppose we make use of a K-dimensional subspace representation (with K quite high, say 20). Then calculating the subspace coefficients for a new image involves K dot-products of full-size feature vectors (of size S, say). Typically, feature vectors might be of size $S = 200 + s$, if we use 200 contour points; and s is the number of scaling levels, minus a few (4, say) noisy levels omitted at the bottom of each CSS image. If $s \simeq 50$, we have a typical feature-vector length $S = 200 + 50 - 4 = 246$. Matching proceeds by forming Euclidean distances from a test feature vector to all objects in the database, for reduced K-vectors, which amounts to forming dot-products. Thus the complexity overall for both creation of the K-vector and for matching amounts to $K \times S + N \times K$ multiplies, if there are N objects in the database.

For the Stage 2 algorithm, with a k-vector for each subspace (say $k \simeq 5$), if there are C classes then we need to form coefficients via a length-S dot-product with k eigenvectors in each class, following by finding magnitudes in the k-space as in eq. (10) above for each class. Thus the number of multiplies is $C \times k \times S + C \times k$.

For example, for the MPEG-7 database (below), $C = 70$, with 20 objects each, making $N = 1400$.

Then the Stage 1 algorithm with $S = 246$ and $K = 20$ takes 32920 multiplies and the Stage 2 algorithm with $k = 5$ takes 86450, amounting to a 2.6-fold increase in complexity for matching.

4 Experiments and Results

The algorithms presented in §3 have been evaluated using three shape databases.

4.1 Test Data Sets

The first database used, shown in Fig. 3, is a set of 131 fish divided into 17 classes of 6 to 8 fish [3]. The original boundary curves in the database have very long parameterization length, ranging from 400 to 1600 points. Here, these are re-parameterized to 200 to speed up execution time.

The second database used comprises 216 shapes, divided into 18 classes: bird, bone, brick, camel, etc. [12]. The original database consists of binary images in TIFF format. We extracted the boundary curves of each image and again parameterized the boundary curve to 200 points. Fig. 5 shows the entire second database. Each row represents one class of object. (A portion of the database has been enlarged and shown in Fig. 1.)

The third database is the MPEG-7 Core Experiment CE-Shape-1 closed-contour database [13]. This consists of C =70 classes (bone, chicken, cellular_ph, etc.) of 20 objects each ($N = 1400$). Again, we made use of length-200 contours.

4.2 Implementation Details

4.2.1 CSS images

We computed the CSS images by smoothing with Gaussians with σ value starting at 5 and incremented by one unit until no zero curvatures remain on the smoothed curve. 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. In order to standardize the size of all CSS images in a database, we padded the y-axis of all CSS images to a constant length, equal to the maximum over all images.

Lengths S of the phase-normalized feature vector (eq. (8)) were 244, 230, and 257 for databases 1, 2, and 3 respectively.

4.2.2 General Eigenspace

We built the Stage 1 eigenspace using the first $K = 15$ SVD basis vectors. In [9], we analyzed the effect of selecting different number of bases and found that there was little effect on results, for $K \geq 3$, with maximum average matching results at about $K = 15$.

We also tested using *raw* CSS images as the vehicle for a subspace reduction [9], with of course a greatly increased execution time. We found that raw images perform substantially worse than marginal-sum feature vectors, likely because of the high noise content in CSS images.

4.2.3 Specialized Eigenspaces

As in §3.3.2, Stage 2 application of the eigenspace method involves creating a specialized eigenspace for each class in a database, and then matching a test vector against each object's eigenspace. We found that this approach produced much better results than matching against of general eigenspace (see §4.4 below). This is not surprising since we now include extra information, the class memebrship. However, we do not match against individual shapes, but just onto whole-class subspaces.

4.2.4 Global parameters

For the *standard algorithm*, a pre-processing step is typically applied to screen out contours that are most likely mismatches for the test contour, over a series of "global" parameters. These are: (1) aspect Ratio a; (2) Eccentricity e; and Circularity c, which we denote the ACE 3-vector.

The Aspect Ratio is the ratio of the number of rows to the number of columns of a CSS image [14]. Since we fix the number of columns to 200, this amounts to a check on the maximum number (unpadded) of scaling levels (rows) in the source and target CSS images. The eccentricity e is the ratio of maximum to minimum eigenvalues of the 2×2 matrix of second-order central moments of the original, 2D curve, an area-based descriptor. In fact, since the matrix is 2D, this can easily be calculated analytically [15]. Circularity c is the ratio of the square of the perimeter to the area, a contour-based parameter. For Stage 2 processing, for each test contour we applied this global parameter screening by asking whether the ACE vector for the test image fell outside the bounding box for all the ACE vectors of an object class.

4.3 Evaluation by Class Matching

For the standard algorithm based on maxima and for Stage 1 eigen-CSS, we evaluated the algorithm using the identical evaluation method described in §2.3. Every boundary curve in a database was matched in turn shape-to-shape against other curves in the database and the matching percentage for a curve, the class of the curves, and the entire database were computed. E.g., Fig. 6 illustrates the matching result for a class 0 fish in database 1. The input fish is shown in the top-left hand corner with the class name labelled over the curve. The rest of the figure depicts the best 15 match results, ranked by their Euclidean distances from the input in the eigenspace subspace. Since the input fish was in the database, it was retrieved as the best matched (it was shown as the first matching result), along with 6 other class-0 fish. Because there are 8 fish in this class, the match percentage for this particular fish was $7/8 * 100\% = 87.5\%$. For all the class-0 fish, the matching average was 73%, and the matching average was 71% for the entire database 1, with no global parameters and a general eigenspace.

4.4 Results

Table 1 shows results for all methods tested. For the fish database, the matching average over all classes was reported to be 76% [3]. In a best-efforts re-implementation as stated above, our figure was 72% (see the top of Table 1). However, our implementation did not test against multiple instances of the same image, since for the method proposed here we did not need to check against images with 180° rotation, since the method is rotation-invariant; also, we did not need to test against mirroring, since as discussed in § 3.2.1 and shown in Fig. 4, the method presented here is reflection-invariant. Hence we simply implemented the "original method" (see [3], pp.77-79) and not the "mirror-image extension" ([3], p.78). As well, other possible improvements mentioned were not implemented. However, we did implement the three ACE "global parameters" recommended (see above in §4.2.4).

In comparison, the Stage 1 algorithm given here also produced a 72% success rate, but with a much simpler algorithm.

However, when Stage 2 was implemented, results gave a 98% success rate, using only $k = 1$ eigenvectors in each class, and measuring success as returning exactly the correct (one) class. Such a large improvement is partly due to the fact that we are now matching to the whole subspace for class-labelled data, not matching to shapes and then evaluating success of categorization. I.e., how Stage 2 proceeds is by finding the best match to class subspaces, not shape-to-shape. Nevertheless, Table 1 uses the same measure — categorization — for all methods.

Using $k = 2$ coefficients, Stage 2 processing achieves 100% classification accuracy for the first two databases for a leave-all-in anaysis. For the MPEG-7 database, the third database tested, we achieved 99% accuracy using only 5 basis vectors. Again note, however, that this result is for class-labelled data, projecting onto class subspaces and not shape-to-shape.

In comparison to previous results reported for the MPEG-7 database [13,16,17], the best result, reported in [17] and using a quite complex hierarchical deformable-shape tree, was a shape-to-shape matching average of 87.7%, for matches in the top 40 objects returned. In comparison, our Stage 1 result for this database is only 61% (see Table 1). However, for Stage 2 we should be comparing to methods that use a shape-to-class matching. Ref. [16] reports a classification rate of 98% using a modified leave-one-out test. In [16], one of the 20 objects in each of 70 classes is removed, and then tested against the 70 classes now consisting of 19 objects each (N=1330). Passing over all take-one-out object indices 1 to 20, there are thus 1400 tests in all. Using the same test, the Stage 2 eigen-CSS method achieved 94% accuracy, using a subspace dimensionality of 10. So the 98% result in [16] is appreciably better. However, the method used there is a considerably more complex algorithm based on contour segments, while the present method does comparably and is very simple. Asking that the correct class appear in the top

2 classes, not the first class returned, Stage 2 eigen-CSS produces a success rate 98.64% at dimension $k = 19$ (Table 1).

Comparing running time, the execution time reported in [16] is 6 sec per classification, in Matlab on a 1.7 GHz machine, whereas the present method (also in Matlab) takes only 7×10^{-5} sec on a 2.8 GHz machine, implying that the present method is about 5 orders of magnitude faster. (As well, our implementation of the original method [3] gives classification timing results about 10^4 times slower, using global parameters, on a dual 2.4 GHz machine.) Applied to range data, the eigen-CSS approach has been shown to achieve substantial speedup with no decrease in accuracy [18]. [3]

5 Discussion and Conclusion

The main advantages of our algorithm are simplicity and execution efficiency. As well, the method possesses the important property of rotation- and reflection-invariance, which obviates searching for multiple instances of objects. The new feature vector also effectively removes noise and produces substantively better results than raw CSS images.

For Stage 1, after building the eigenspace, which has to be performed only once for a database, matching an image against the models in a database involves mapping the image to the eigenspace and calculating the distances between the image and the models in the database. There is no complicated searching, as in the canonical method [3]. With the length of the feature vector as small as 15 elements, the matching is extremely efficient. However, for the MPEG-7 database Stage 1 processing is considerably weaker than the best shape-to-shape methods available [17].

For Stage 2, wherein multiple eigenspaces are formed, the method is still efficient, and performs well for the MPEG-7 database. For that database, we found that leaving all elements in, when building the subspaces, achieves essentially 100% accuracy.

Moreover, for the leave-one-out test as well, suppose we perform mean-subtraction on all vectors, subtracting a candidate class mean from a test vector before matching to class subspaces using mean-subtraction, we again achieve 100% accuracy provided the mean is constructed using all class members. Therefore this indicates that the variability in a subspace is substantial, and a single member can be quite different from the rest of a class. If a larger number of instances of a class is available, then leave-all-in or leave-one-out tests should have about the same success rate. And in fact we tried a complete leave-one-out test on a set of classes of similar and overlapping x, y curves, but in C families. For 26-dimensional curves and 100 objects in each training class, for Stage 2 processing leave-one-out or leave-all-in makes essentially no difference. This is because a single object out of 100 cannot greatly shift the subspace. Therefore for larger sets of objects in each class, we expect our algorithm to do better. Other techniques that also increase accuracy include using an L1 norm either for subspace coefficients or full vectors, and also shifting every feature vector to have individual mean of zero: results reported in Table 1 for the

[3] Note that Ref. [18], which uses [9] and Stage 2, appeared after the Stage 2 version of this paper was submitted (April, 2006).

MPEG7 leave-one-out test make use of mean-subtraction and L1 norm, since these techniques make a (very slight) improvement in results. We also tested using an Independent Component Analysis (ICA) basis instead of an SVD basis, but SVD gave marginally better results than ICA.

We have also applied the method explicated here to the problem of recognizing 3D objects by their boundary shape [19]. Testing on a 3D database comprising 209 colour objects in 2926 different view poses, the algorithm achieves a 97% recognition rate for object type and 95% for object pose. The success of the method in quite a difficult problem domain amply justifies the suitability of the algorithm set out here. In sum, the method we present here is simple, fast, and effective.

References

[1] F. Mokhtarian, A. Mackworth, Scale-based description and recognition of planar curves and two-dimensional shapes, IEEE Trans. Patt. Anal. and Mach. Intell. 8 (1986) 34–43.

[2] F. Mokhtarian, Silhouette-based isolated object recognition through curvature scale space, IEEE Trans. Patt. Anal. and Mach. Intell. 17 (1995) 539–544.

[3] F. Mokhtarian, M. Bober, Curvature Scale Space Representation: Theory, Applications, and MPEG-7 Standardization, Kluwer, 2003.

[4] H. Murase, S. Nayar, Learning object models from appearance, in: 11th National Conf. on Artificial Intelligence: AAAI93, 1993, pp. 836–843.

[5] H. Murase, S. Nayar, Illumination planning for object recognition in structured environments, in: Computer Vision and Patt. Recog.: CVPR94, 1994, pp. 31–38.

[6] M. Drew, J. Wei, Z. Li, Illumination–invariant color object recognition via compressed chromaticity histograms of color–channel–normalized images, in: ICCV98, IEEE, 1998, pp. 533–540.

[7] C. Kuglin, D. Hines, The phase correlation image alignment method, in: Int'l Conf. on Cybernetics and Society, 1975, pp. 163–165.

[8] S. Abbasi, F. Mokhtarian, Affine-similar shape retrieval: Application to multi-view3-D object recognition, IEEE. Trans. on Image Proc. 10 (2001) 131–139.

[9] M. S. Drew, T. K. Lee, A. Rova, Shape retrieval with eigen-CSS search, Tech. Rep. TR 2005-7, Simon Fraser University, http://fas.sfu.ca/pub/cs/TR/2005/CMPT2005-07.pdf (2005).

[10] K. Arbter, W. Snyder, H. Burkhardt, G. Hirzinger, Application of affine invariant Fourier descriptors to recognition of3-D objects, IEEE Trans. Patt. Anal. and Mach. Intell. 12 (1990) 640–647.

[11] I. Kunttu, L. Lepist o, J. Rauhamaa, A. Visa, Recognition of shapes by editing shock graphs, in: Int. Conf. on Computer Vision: ICCV2001, 2001, pp. 755–762.

[12] T. Sebastian, P. Klein, B. Kimia, Recognition of shapes by editing their shock graphs, IEEE Trans. Patt. Anal. and Mach. Intell. 26 (2004) 550–571.

[13] L. J. Latecki, R. Lakämper, U. Eckhardt, Shape descriptors for non-rigid shapes with a single closed contour, in: Computer Vision and Patt. Recog.: CVPR2000, 2000, pp. 1424–1429.

[14] F. Mokhtarian, H. Murase, Silhouette-based isolated object recognition through curvature scale space, in: Int. Conf. on Computer Vision: ICCV93, 1993, pp. 269–274.

[15] M. Drew, Z.-N. Li, Z. Tauber, Illumination color covariant locale-based visual object retrieval, Patt. Rec. 35 (8) (2002) 1687–1704.

[16] K. Sun, B. Super, Classification of contour shapes using class segment sets, in: CVPR05, 2005, pp. II: 727–733.

[17] P. Felzenszwalb, J. D. Schwartz, Hierarchical matching of deformable shapes, in: Computer Vision and Patt. Recog.: CVPR2007, 2007, pp. 1–8.

[18] S. Stiene, K. Lingemann, A. Nuchter, J. Hertzberg, Contour-based object detection in range images, in: 3DPVT: 3rd Int. Symp. on 3D Data Proc., Visualization, and Transmission, IEEE, June 14-16, 2006, pp. 168–175.

[19] T. Lee, M. Drew, 3d object recognition by scale-space of contours, in: First Int. Conf. on Scale Space Methods and Variational Methods in Computer Vision (SSVM'07), 2007, pp. 883–894.

Algorithm	#Basis Vectors	#Best Matches	GlobalParam's	Shapes/Classes	Average %
Database 1 (Fish): 131 Objects, 17 Classes					
Maxima		15 objects	No	S	62
		15 objects	Yes	S	72
General	15	15 objects	No	S	71
Eigen-CSS	15	15 objects	Yes	S	72
Specialized	1	1 top class	No	C	98
Eigen-CSS	2	1 top class	No	C	100
Database 2 (Sebastian et al.): 216 Objects, 18 Classes					
Maxima		15 objects	No	S	68
		15 objects	Yes	S	74
General	15	15 objects	No	S	72
Eigen-CSS	15	15 objects	Yes	S	75
Specialized	1	1 top class	No	C	97
Eigen-CSS	2	1 top class	No	C	100
Database 3 (MPEG-7): 1400 Objects, 70 Classes					
Maxima		15 objects	No	S	3
		15 objects	Yes	S	12
		40 objects	No	S	5
		40 objects	Yes	S	50
General	15	15 objects	No	S	37
Eigen-CSS	15	40 objects	No	S	56
	15	40 objects	No	S	61
Specialized	1	1 top class	No	C	84
Eigen-CSS	2	1 top class	No	C	93
	5	1 top class	No	C	99
	17	1 top class	No	C	100
Specialized	11	1 top class	No	C	86
Eigen-CSS,	10	1 top class	Yes	C	94
Leave-1-out	19	2 top classes	Yes	C	99

Table caption:
Matching averages for databases 1, 2, and 3. The standard method, “Maxima”, and the Stage 1 Eigen-CSS method both examine how many objects in the top 15 returned are in the correct class. Stage 2 “Specialized Eigen-CSS” asks which are the top classes returned. Under “Shapes/Classes”, an “S” indicates that we match shape-to-shape and calculate class-categorization accuracy as in §2.3; and “C” means we are finding the best match for a low-D vector amongst the class subspaces. We are interested in using the fewest number of basis coefficients to find the correct class. Leaving all data in tests, here we report results using only a few basis vectors, and ask for results only in the single top classification returned. The best classification result is the Specialized Eigen-CSS method, using about 17 basis vectors with a resulting accuracy of 99.8%. For a leave-one-out test, the Stage 2 processing achieves 86% accuracy using 11 basis vectors, when asking for the method to return the correct class as #1, When global parameter 3-vector ACE screening is applied, Stage 2 processing achieves 94% accuracy, for dimensionality 10. Asking that the correct class appear in the top-2 classes yields 99% accuracy for dimensionality 19. The considerably more complex method set out in [16] achieved accuracy of 98% for the leave-one-out MPEG7 test, but has running time 5 orders of magnitude longer.

Figure Captions

1. Sample of boundary curves.
2. (a): Gaussian smoothing process of a closed curve shown at the leftmost figure. (b): The corresponding curvature scale space image.
3. A database with 131 fish divided into 17 classes. Every row represents a class of fish (see [3].)
4. Mirror reflection. (a): The boundary curve and its $180°$ rotation and vertical mirror transformations. (b): The corresponding CSS images. (c): The corresponding vectors $\mathbf{r}$ as computed by eq. (5). (d): The corresponding phase-normalized vectors $\tilde{\mathbf{r}}$.
5. A shape database of 216 boundary curves, divided into 18 classes. Every row represents a class of object.
6. Matching results for a fish in database 1, using our algorithm. The top-left curve shows the input fish. The rest of the fish show the best 15 match results, ranked by their Euclidean distance to the input fish in the eigenspace subspace. The fish's class appears as a label over each fish.

Figures

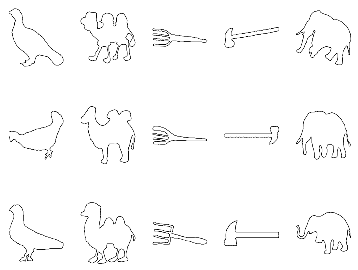

Fig. 1. Sample of boundary curves.

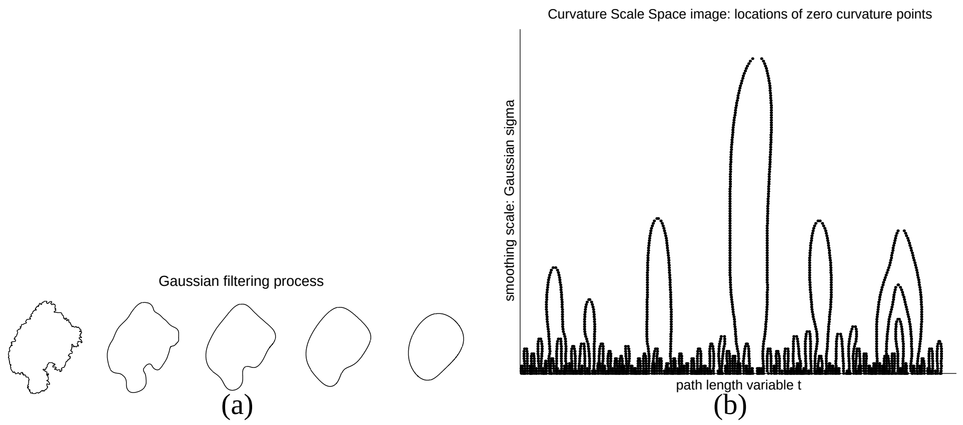

Fig. 2. (a): Gaussian smoothing process of a closed curve shown at the left most figure. (b): The corresponding curvature scale space image.

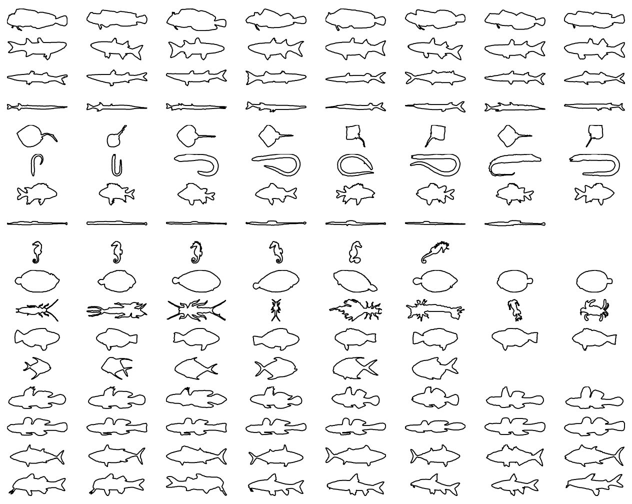
Fig. 3. A database with 131 fish divided into 17 classes. Every row represents a class of fish (see [3].)

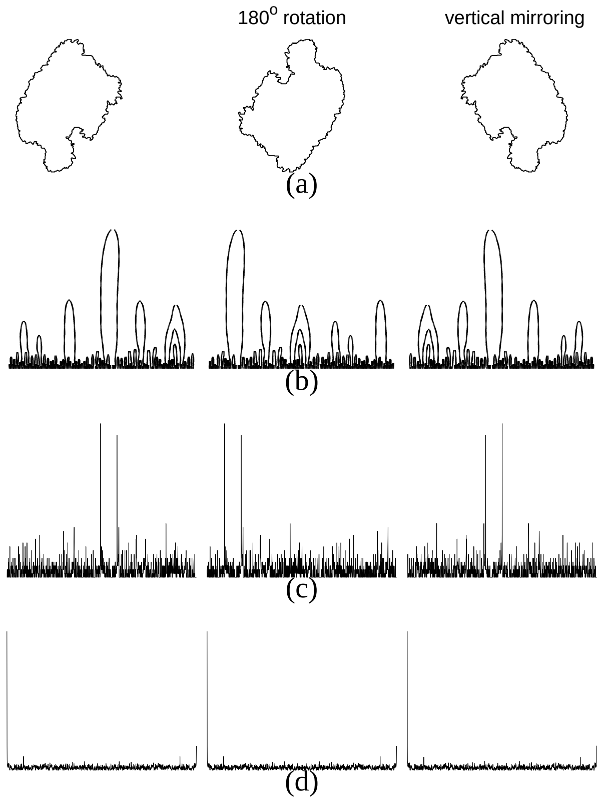

Fig. 4. Mirror reflection. (a): The boundary curve and its 180° rotation and vertical mirror transformations. (b): The corresponding CSS images. (c): The corresponding vectors $\mathbf{r}$ as computed by eq. (5). (d): The corresponding phase-normalized vectors $\tilde{\mathbf{r}}$.

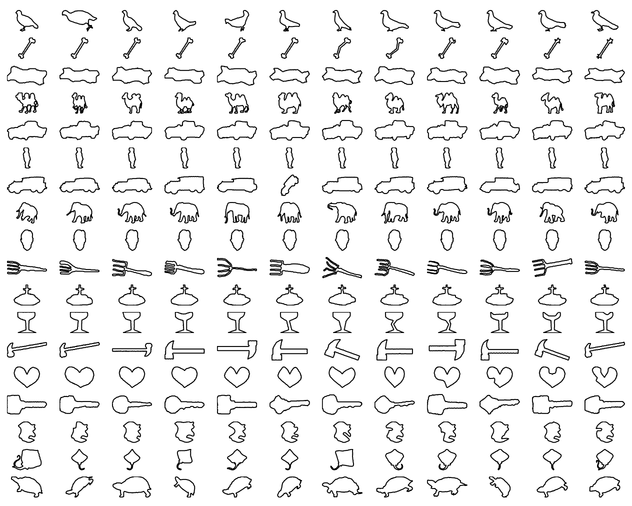

Fig. 5. A shape database of 216 boundary curves, divided into 18 classes. Every row represents a class of object.

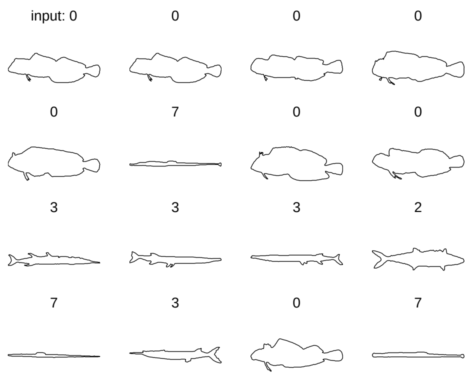

Fig. 6. Matching results for a fish in database 1, using our algorithm. The top-left curve shows the input fish. The rest of the fish show the best 15 match results, ranked by their Euclidean distance to the input fish in the eigenspace subspace. The fish's class appears as a label over each fish.