PERGAMON

Pattern Recognition 35 (2002) 31–41

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Shape similarity retrieval under affine transforms

Farzin Mokhtarian*, Sadegh Abbasi

Centre for Vision Speech and Signal Processing, Department of Electronic & Electrical Engineering, University of Surrey, Guildford GU2 7XH, UK

Abstract

The maxima of curvature scale space (CSS) image have already been used to represent 2-D shapes in different applications. The representation has shown robustness under the similarity transformations. Scaling, orientation changes, translation and even noise can be easily handled by the representation and its associated matching algorithm. In this paper, we examine the robustness of the representation under general affine transforms. We have a database of 1100 images of marine creatures. The contours in this database demonstrate a great range of shape variation. A database of 5000 contours has been constructed using 500 real object boundaries and 4500 contours which are the affine transformed versions of real objects. The CSS representation is then used to find similar shapes from this prototype database. The results provide substantial evidence of stability of the CSS image and its contour maxima under affine transformation. The method is also evaluated objectively through a large classified database and its performance is compared with the performance of two well-known methods, namely Fourier descriptors and moment invariants. The CSS shape descriptor has been selected for MPEG-7 standardization.

Keywords: Multi-scale analysis; Shape similarity retrieval; Curvature scale space; Affine transform; Affine length

1. Introduction

Considerable amount of information exists in two dimensional boundaries of objects which enables us to recognise objects without using further information. However, despite great effort [1], the problem of shape representation in computer vision is still a very difficult one, and remains so in shape similarity retrieval [2–7].

A shape is originally defined by x and y coordinates of its boundary points which is subject to change if the distance of camera and object changes or the object is rotated in front of a fixed camera or the origin is altered. As a result, a shape representation must be robust under similarity transformation which includes scaling, changes in orientation and translation. If the camera is allowed to change its viewpoint with respect to the object, the resulting boundary of the object will be deformed. The deformation can be approximated by general affine transforms.

A number of shape representations have been suggested to recognise shapes even under affine transformation. Some of them are the extensions of well-known methods such as Fourier descriptors [8] and moment invariants [9–11]. The methods are then tested on a small number of objects for the purpose of pattern recognition. In both methods, the basic idea is to use a parametrisation which is robust with respect to affine transformation. The arc length representation is not transformed linearly under general affine transform and therefore is replaced by *affine length* [12,13]. In all cases, the methods are tested on a small number of objects and therefore the results are not reliable. Moreover, almost the same results can be achieved using conventional methods without modifications [8,10].

Affine invariant scale space is introduced in Refs. [14,15]. It generalises the definition of curvature and introduces *affine curvature* (see also Ref. [16]). This curve evolution method is proven to have similar

* Corresponding author. Tel.: +44-1483-876035; fax: +44-1483-876031.

E-mail addresses: f.mokhtarian@surrey.ac.uk (F. Mokhtarian), s.abbasi@ee.surrey.ac.uk (S. Abbasi).

0031-3203/01/$20.00
PII: S0031-3203(01)00040-1

properties as curvature evolution [17–19] as well as being affine-invariant. However, an explicit shape representation has yet to be introduced based on the theory of affine invariant scale space [20]. The prospective shape representation might be computationally complex as the definition of affine curvature involves higher order derivatives.

We have already used the maxima of curvature scale space (CSS) image to represent shapes of boundaries in similarity retrieval applications [21,22]. The representation is proved to be robust under similarity transformation which include translation, scaling and changes in orientation. In this paper, we examine the robustness of the representation under general affine transforms.

The CSS representation finds its roots in curve evolution [23,24] curvature deformation and heat equation. In fact, the *resampled* curvature scale space [25] implements curvature deformation [17]. This is carried out by convolving each coordinate of a closed planar curve, with a Gaussian function at different levels of scale. At each stage and before being convolved with a larger width Gaussian, the curve is represented in terms of arc length parameter. In *regular* curvature scale space [25] the resampling is not applied. As a result, the process is not a curvature deformation anymore. However, the implementation is carried out much faster and the representation has shown a good performance in object recognition [26], and shape similarity retrieval [22,27].

The CSS image representation employs the arc length parametrisation which is not affine invariant. As a result, we expect some deviation in the maxima of the CSS image under general affine transformation. It has been shown that affine invariance can only be achieved by an affine invariant parametrisation and affine length has been used by a number of authors [8–10,14]. We also examine the utility of using affine length instead of arc length to parametrise the curve prior to computing its CSS image.

We have a database of 1100 images of marine creatures. The contours in this database demonstrate a great range of shape variation. A database of 5000 contours has been constructed using 500 real object boundaries and 4500 contours which are the affine transformed versions of real objects.

The following is the organisation of the remainder of this paper. In Section 2 the background of the method is reviewed. Curvature measurement, curve smoothing and construction of the CSS image are explained in this section. Section 3 briefly discusses the properties of the CSS image. Section 4, is concerned with the matching algorithm. We use simple examples to explain how two sets of CSS maxima are compared. Section 5 is about the affine transforms and procedure we follow to create our large databases. In Section 6 the experimental results are presented. We evaluate the performance of the method in two different ways. The results are then compared to the results of other well-known methods in Section 7. The concluding remarks are presented in Section 8.

2. The CSS image

This section describes some aspects of the CSS image. It commences by explaining the process of the CSS image construction. Then some properties of the representation are reviewed.

2.1. Curvature

Consider a parametric vector equation for a curve

$$\vec{\Gamma}(u) = (x(u), y(u)),$$

where u is an arbitrary parameter. The formula for computing the curvature function is expressed as

$$\kappa(u) = \frac{\dot{x}(u)\ddot{y}(u) - \ddot{x}(u)\dot{y}(u)}{(\dot{x}^2(u) + \dot{y}^2(u))^{3/2}}. \tag{1}$$

Curve smoothing prior to curvature measurement reduces the effects of noise. The computation starts by convolving each coordinate of the curve with a Gaussian function. In continuous form we have

$$X(u,\sigma) = x(u)\bigstar g(u,\sigma),$$
$$Y(u,\sigma) = y(u)\bigstar g(u,\sigma). \tag{2}$$

where $\bigstar$ denotes convolution. According to the properties of convolution, the derivatives of every component can be calculated easily

$$\dot{X}(u,\sigma) = x(u)\bigstar \dot{g}(u,\sigma),$$
$$\ddot{X}(u,\sigma) = x(u)\bigstar \ddot{g}(u,\sigma) \tag{3}$$

and we have similar formulas for $\dot{Y}(u,\sigma)$ and $\ddot{Y}(u,\sigma)$. Since the exact forms of $\dot{g}(u,\sigma)$ and $\ddot{g}(u,\sigma)$ are known, the curvature of the smoothed curve can be computed as

$$\kappa(u,\sigma) = \frac{\dot{X}(u,\sigma)\ddot{Y}(u,\sigma) - \ddot{X}(u,\sigma)\dot{Y}(u,\sigma)}{(\dot{X}^2(u,\sigma) + \dot{Y}^2(u,\sigma))^{3/2}}. \tag{4}$$

If u is the arc length parameter, $\vec{\Gamma}(u)$ is called the *natural representation* of the curve. However, even if $\vec{\Gamma}$ is the natural representation of the original curve, in general

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

is not a natural representation of the smoothed curve. The main feature of this method is that in digital form, the adaptation of a definition for first and second derivatives is not needed and curvature along the smoothed digital curve can be calculated directly from Eq. (4). In order to do this, the derivatives of Gaussian are first sampled at M points, where M is the nearest odd number to 10σ. Then Eq. (3) is implemented as follows:

$$\dot{X}(n,\sigma) = \sum_{k=-L}^{L} x(n-k)\dot{g}(k,\sigma),$$

$$\ddot{X}(n,\sigma) = \sum_{k=-L}^{L} x(n-k)\ddot{g}(k,\sigma),$$

where $L = (M-1)/2$ is an even number. Similar equations are used to compute $\dot{Y}(n,\sigma)$ and $\ddot{Y}(n,\sigma)$.

2.2. Construction of the CSS image

Following the preprocessing stage, every object is represented by the x and y coordinates of its boundary points. The number of these points varies from 400 to 1200 for images in our prototype databases.

To normalise the arc length, the boundary is resampled and represented by 200 equally distant points. This will be the natural representation of the boundary.

Considering the resampled curve as

$$\vec{\Gamma}_0(s) = (x_0(s), y_0(s)),$$

we smooth the curve by Gaussian function as described before

$$X(s,t) = x_0(s) \bigstar g(s,t), \quad Y(s,t) = y_0(s) \bigstar g(s,t).$$

The smoothed curve is called $\vec{\Gamma}_\sigma$, where σ denotes the width of the Gaussian kernel. It is then possible to find the locations of curvature zero crossings on Γ_σ by using equation (4). The process starts with $\sigma = 1$, and at each level, σ is increased by $\Delta\sigma$, chosen as 0.1 in our experiments. As σ increases, Γ_σ shrinks and becomes smoother, and the number of curvature zero crossing points on it decreases. Finally, when σ is sufficiently high, Γ_σ will be a convex curve with no curvature zero crossings (see Fig. 1(a)). The process of creating ordered sequences of curves is referred to as the *evolution* of Γ.

If we determine the locations of curvature zero crossings of every Γ_σ during evolution, we can display the resulting points in (u,σ) plane, where u is an approximation of the normalised arc length and σ is the width of the Gaussian kernel. The result of this process can be represented as a binary image called *the regular CSS image* of the curve (see Fig. 1(b)). The intersection of every horizontal line with the contours in this image indicates the locations of curvature zero crossings on the corresponding evolved curve Γ_σ. For example, by drawing a horizontal line at $\sigma = 10.0$, it is observed that there are 6 zero crossing points on Γ_{10}. These points can also be found on the boundary of object in Fig. 1(a) for $\sigma = 10$.

From Fig. 1(a), we learn that on each concavity of the shape there are two curvature zero crossings, and as the curve is smoothed they approach each other and finally when the concavity is filled, they join. The pair of curvature zero crossings create a contour in the CSS image which represents the relevant concavity.

2.3. Extracting maxima of CSS contours

We represent every image in the database with the locations of its major CSS contour maxima. For example, in Fig. 1(b) there are seven major maxima, and therefore the image will be represented by seven pairs of integer numbers. The locations of maxima are not readily available and must be extracted from the image [22].

Small contours of the CSS image are related to noise or small ripples of the curve. In order to avoid complicated and inefficient matching, small maxima are not included in the representation. In our system, if a maximum is less than $\frac{1}{6}$ of the largest maximum of the same CSS image, it is considered as noise. As a result, only major concavities and convexities of a shape will contribute to the representation.

3. Properties of the CSS image

The CSS representation is robust with respect to scale, noise and change in orientation. A rotation of the object usually causes a circular shift in its representation which is easily determined during the matching process. Note that the effect of a change in the starting point is also the same. Noise may create some small contours in the CSS image, but the main contours and therefore the corresponding maxima remain unaffected.

Compactness is another feature of the CSS representation. A shape is represented by less than ten pairs of

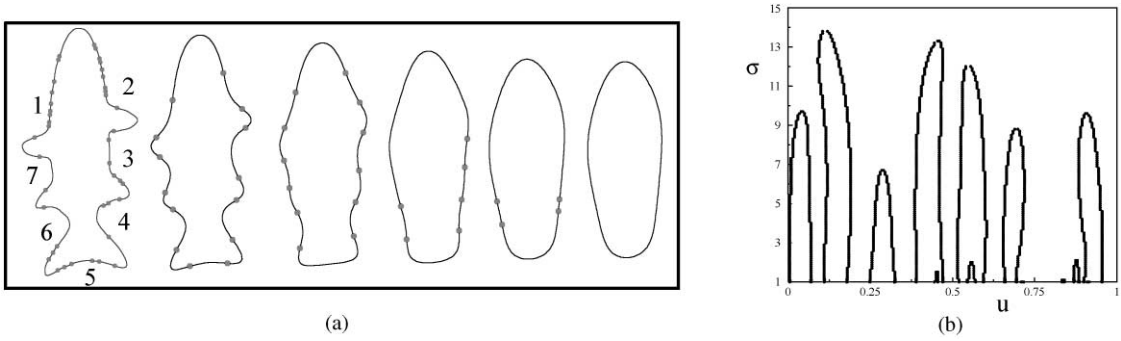

Fig. 1. (a) Shrinkage and smoothing of the curve and decreasing of the number of curvature zero crossings during the evolution, from left: $\sigma = 1, 4, 7, 10, 12, 14$. (b) The CSS image of the shape.

integer values which can be determined without any ambiguity. The matching algorithm which compares two sets of representations and assigns a match value as the measure of similarity between the shapes is also simple and fast.

Another property of the CSS image is that it retains the local information of the shape. Every contour of the CSS image corresponds to a concavity or a convexity of the shape. A local deformation of the shape mainly causes a change in the corresponding contour of the CSS image. Using this property, one can include more local information about the shape in the CSS image.

3.1. CSS under affine transforms

We now show that the curvature zero crossing points are preserved under affine transformation. As a result, the overall configuration of the CSS image is also preserved. This can also be verified by recalling the fact that the number and orders of the shape segments remain unchanged under general affine transformation. We then verify these facts through several examples.

The general affine transformation can be represented mathematically with the following equations:

$$x_a(u) = ax(u) + by(u) + e,$$

$$y_a(u) = cx(u) + dy(u) + f, \tag{5}$$

where $x_a(u)$ and $y_a(u)$ represent the coordinates of the transformed shape. There are six degrees of freedom in this transformation. Translation and uniform scaling are represented by two degrees of freedom, while change in orientation needs just one parameter. The remaining parameter is related to *shear*. The general affine transformation contains all these transformations.

Theorem. *If Γ_a is the transformed version of a planar curve Γ under general affine transformation, there is a one to one correspondence between the curvature zero crossings of Γ and Γ_a.*

Proof. From Eq. (5) we have

$$\dot{x}_a(u) = a\dot{x}(u) + b\dot{y}(u), \quad \dot{y}_a(u) = c\dot{x}(u) + d\dot{y}(u)$$

Likewise,

$$\ddot{x}_a(u) = a\ddot{x}(u) + b\ddot{y}(u), \quad \ddot{y}_a(u) = c\ddot{x}(u) + d\ddot{y}(u)$$

And therefore from Eq. (1)

$$\kappa_a(u) = \frac{(a\dot{x}(u) + b\dot{y}(u))(c\ddot{x}(u) + d\ddot{y}(u))}{((a\dot{x}(u) + b\dot{y}(u))^2 + (c\dot{x}(u) + d\dot{y}(u))^2)^{3/2}} - \frac{(a\ddot{x}(u) + b\ddot{y}(u))(c\dot{x}(u) + d\dot{y}(u))}{((a\dot{x}(u) + b\dot{y}(u))^2 + (c\dot{x}(u) + d\dot{y}(u))^2)^{3/2}}$$

and then

$$\kappa_a(u) = \frac{(ad - bc)(\dot{x}\ddot{y} - \dot{y}\ddot{x})}{((a\dot{x}(u) + b\dot{y}(u))^2 + (c\dot{x}(u) + d\dot{y}(u))^2)^{3/2}}. \tag{6}$$

Now compare the numerators of Eqs. (1) and (6) and observe that the curvature zero crossings of the original and the transformed shapes have a one to one correspondence. □

As this theorem shows, in all cases, even those with severe deformations, the number of the CSS contours are preserved. The order of shape segments is also preserved under general affine transforms. As a result, the number and order of the CSS contours are preserved. We observed that even the height of the corresponding CSS contours are not very different. However, small changes in the locations of maxima are inevitable. As a result of the changes in the CSS maxima, the matching value between a shape and its transformed version is not zero.

4. The underlying ideas in CSS matching

We assume that the user enters his query by sketching a boundary of his desired object or by pointing to an image. In each case, we carry out the same preprocessing to find the maxima of the CSS contours of the input image and compare them with the same descriptors of the database images. For convenience, from now on, we call the input as *image* and the images in the database as *models*.

After extracting the maxima of every model, we normalise their coordinates by dividing them by the number of samples, e.g. 200; so that the horizontal coordinate u varies in the range [0,1]. This will ensure that the comparison is meaningful even if the number of samples in the image and the model are different. The maxima of every model are sorted according to their σ-coordinates during the process of maxima extraction.

In this section we explain the basic concepts of our matching algorithm which compares two sets of maxima and assigns a matching value to them. This value is used as a measure of similarity between the actual boundaries of objects. A complete description of the CSS matching algorithm can be found in Ref. [28].

We start with a very simple example to explain the basic concepts. Two CSS images are presented in Fig. 2 and we wish to determine how similar they are. Both CSS images have three contours; however, the locations of the contours do not match. Does it mean that the CSS images are not similar?

Since applying a *circular shift* to the CSS image is equivalent to a *change in orientation* of the corresponding object, we can always apply a circular shift to a CSS image during the matching process to find the best

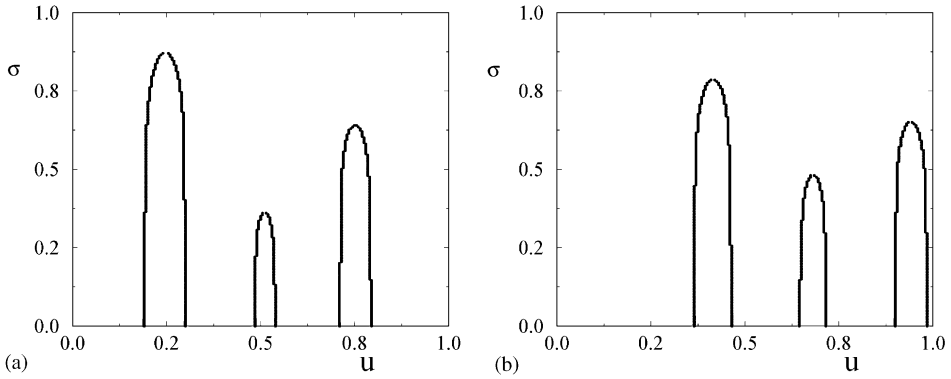

Fig. 2. First example of two CSS images which must be matched. Note that we only use the maxima of CSS contours in the matching process. The whole CSS images are shown in this Figure for a better understanding.

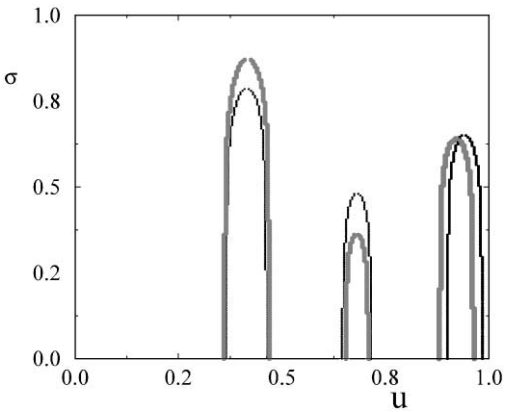

Fig. 3. In this example, the best match is achieved by matching the largest maximum of the image with the largest maximum of the model. The summation of Euclidean distances between the corresponding maxima is the matching cost.

match. In fact by doing this, we change the orientation of the corresponding shape to find the best match between the two shapes.

Unfortunately, the optimum shift which leads to the best match is unknown. Ideally, all possible shifts should be examined to find the best match; but this is very time consuming and inefficient. However, it appears that *a good candidate for the best shift* is the one which makes the u_coordinate of the largest maximum of the image equal to the u_coordinate of the largest maximum of the model. We examine this hypothesis for our example in Fig. 3 and observe that the result is promising. The two sets of maxima are very close to each other.

After applying the shift, we are in a position to measure the similarity between the two CSS images. In order to do this, we have to find for each maximum of the image, the corresponding maximum of the model. Since the two largest maxima have already been matched, we start the process with the second largest maximum of the image and calculate its Euclidean distances with all remaining model maxima. It is then matched with the maximum of the model that is closest in Euclidean distance. Then we match the third maximum of the image with one of the remaining model maxima using the same criteria. This process continues until no image maxima are left. The *final total cost of the match* is the summation of Euclidean distances between the matched maxima. Note that if the number of image maxima differs from the number of individual model maxima, some of the maxima remain unmatched. The height of these maxima (σ_coordinates) is added to the cost of the total match.

As mentioned before, every CSS maximum relates to a concavity or a convexity of the corresponding shape and the u_coordinate of the maximum reflects the location of the relevant segment on the shape. During our experiments, we discovered that the difference between the u_coordinates of a pair of matched maxima should not be too large, e.g. more than 0.2 of the maximum possible distance. Therefore, an image maximum is left unmatched if the difference between its u_coordinate and the u_coordinate of its nearest model maxima is more than a threshold. This ensures that the relevant matched segments are almost in the same position on the shapes. The height of the unmatched image maximum is added to the matching cost.

For the previous example, the best shift was the one which resulted in the match between the two largest maxima, one from the image and the other from the model. But is this always true? As shown in the second example (see Fig. 4), this is not always the case.

As seen in Fig. 4b, the largest maximum of the model is close to the second and even the third largest maximum. As a result, the above mentioned shift may not lead to the best match and we may need to consider several hypotheses as alternatives and find the best match amongst them. Each of these *hypotheses* includes a shift which matches the largest maximum of the image with the first, the

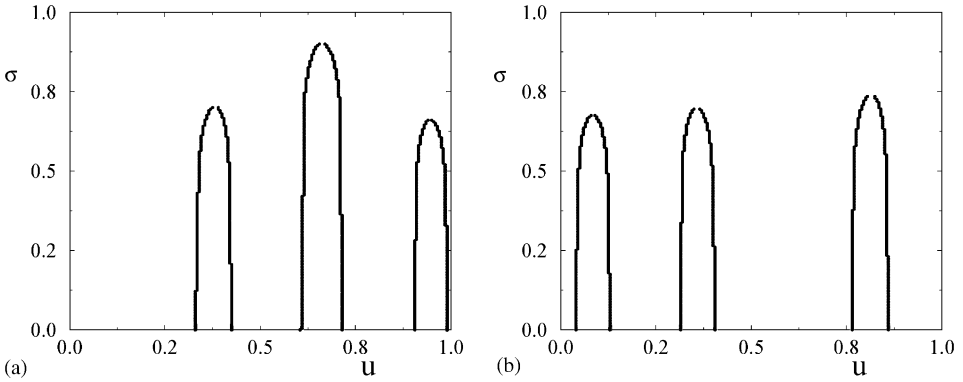

Fig. 4. Second example. (a) Image. (b) Model.

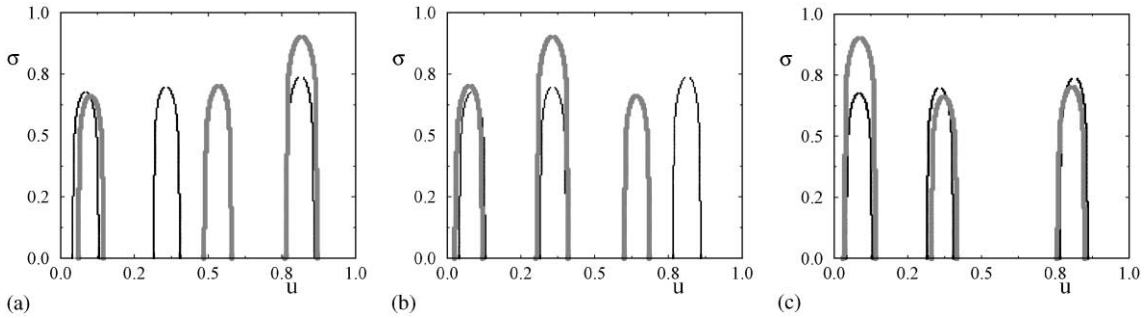

Fig. 5. Different hypotheses for the second example. Note that only the maxima of CSS contours are matched. The whole CSS images are shown in this figure for a better understanding. (a) Hypothesis one, matches the largest maximum of the image with the largest maximum of the model. (b) Hypothesis two, matches the largest maximum of the image with the second largest maximum of the model. (c) Hypothesis three, matches the largest maximum of the image with the third largest maximum of the model.

second and possibly the third largest maximum of the model. These are shown in Fig. 5. Note that several other hypotheses can also be considered. For example, they may include hypotheses which matches the second largest maximum of the image with the first, the second and the third largest maximum of the model.

For each case, the corresponding maxima and the matching cost should be determined. The lowest matching cost will then represent the best match and will be chosen as a measure of similarity between the two sets of the CSS maxima. If the number of hypotheses is relatively large, it is worth using the *best-first* matching strategy [29]. In this method, instead of finding the matched maxima and matching cost for every single hypothesis, it is done gradually and in parallel for all of them until the lowest-cost complete match is found.

5. General affine transforms and our databases

Although the general affine transform in Eq. (5) contains translation, scaling, change in orientation and shear, it is possible to apply only one of these transforamtions at a time. Scaling, change in orientation and shear are represented by the following matrices:

$$A_{scaling} = \begin{pmatrix} S_x & 0 \\ 0 & S_y \end{pmatrix}, \quad A_{rotation} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix},$$

$$A_{shear} = \begin{pmatrix} 1 & k \\ 0 & 1 \end{pmatrix}.$$

If S_x is equal to S_y, $A_{scaling}$ represents a uniform scaling. A shape is not deformed under rotation, uniform scaling and translation. However, nonuniform scaling and shear contribute to the shape deformation under general affine transformation. We examine the performance of the CSS representation under shear transform. The measure of shape deformation depends on the parameter k, *shear ratio*, in the matrix A_{shear}. In the present form of the matrix A_{shear}, x axis is called *shear axis*, as the shape is pulled toward this direction.

Fig. 6 shows the effects of affine transformation on shape deformation. In this Figure, shear ratio is selected

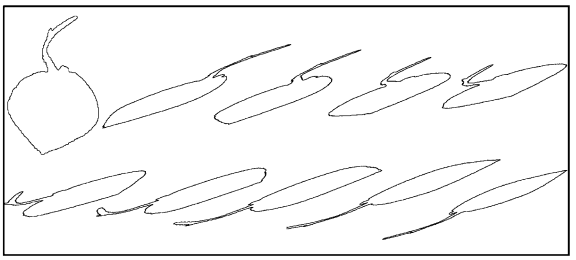

Fig. 6. The deformation of shapes is considerable even with $k = 1$ in shear transform. The original shape is presented in top left. Others represent transformation with $k = 1$ and $\theta = 20°, 40°, \ldots, 160°, 180°$.

as $k = 1$. In order to achieve different shear axes, we have changed the orientation of the original shape prior to applying the pure shear transformation. The values of θ range from 20° to 180°, with 20° intervals. As this figure shows, the deformation is severe for $k = 1$. For larger values of k, e.g. 1.5 and 2, the deformation is much more severe.

In order to create different databases, we choose different values for shear ratio, 1.0, 2.0 and 3.0. We then apply the transformation to a database of 500 original object contours. From every original object, we obtain 9 transformed shapes with different values of θ. Therefore, each database consists of 500 original and 4500 transformed shapes. We then carry out a series of experiments on these databases to verify the robustness of the CSS image representation under affine transformations. The following sections are concerned with these experiments.

5.1. Affine length

In order to construct the CSS image of a planar curve, it is first represented by its normalised arc length parameter. The normalised arc length parameter can be calculated using the following formula:

$$s = \frac{\int_0^u (\dot{x}^2 + \dot{y}^2)^{1/2}}{\int_0^1 (\dot{x}^2 + \dot{y}^2)^{1/2}}.$$

It is obvious that s is not preserved under the transformation represented by Eq. (5). As a result, the normalised arc lengths of the corresponding points on the original and the transformed shapes differ. In order to achieve an affine invariant parametrisation, arc length is usually replaced by affine length which has the following definition.

$$\tau = \frac{\int_0^u (\dot{x}\ddot{y} - \ddot{x}\dot{y})^{1/3}}{\int_0^1 (\dot{x}\ddot{y} - \ddot{x}\dot{y})^{1/3}}.$$

The main disadvantage of affine length is that its computation requires higher order derivatives. However, by using the method described in Section 2, we can parametrise the curve using this formula.

The CSS images can be reconstructed using affine length instead of arc length. In the regular CSS image, only the initial representation is affine length and reparametrisation is not applied as the curve is smoothed. In the following section, we present the results of our experiments which confirms the advantage of using affine length instead of arc length.

6. Experiments and results

We examined the performance of the representations through two different experiments. The first one was performed on the databases described in Section 5. Every original shape was selected as the input query and the first n outputs of the system were observed to see if the transformed versions of the query are retrieved by the system. Both regular and resampled CSS representations with arc length and affine length parametrisation were examined. The results indicated that using affine length parametrisation instead of arc length improves the performance of both representations.

Considering each original, i.e. not affine transformed, shape as an input query, we observed the first n outputs of the system and determined m, the number of outputs which are the affine transformed versions of the input. The success rate for a particular input is calculated as follows:

$$\text{Success rate for an input query} = \frac{m}{m_{max}} \times 100, \qquad (7)$$

where m_{max} is the maximum possible value of m. Note that m_{max} is equal to n if $n \leqslant 10$; if not, m_{max} is equal to 10. The success rate of the system for the whole database will be the average of the success rates for each input query.

It is obvious that the success rate is a function of n, the number of observed outputs. If n is large, it is more likely that all 10 affine transformed versions of the input query appear among the outputs. On the other hand; the first few outputs of the system almost always are the transformed versions of the input. As a result; if n is small, then the success rate is large. The success rate is also a function of k, the shear ratio. The larger values of k result in more deformation and perhaps lower success rate.

We chose different values for n, ranging from 2 to 40, and in each case found the average success rate of the system for *all* 500 original shapes. The same experiment was carried out on four different CSS representations, including regular and resampled CSS image with arc or affine length parametrisation.

The results are presented in Fig. 7(a) and (b). Each figure includes three curves associated with three values of k, the shear ratio. Each curve shows the average

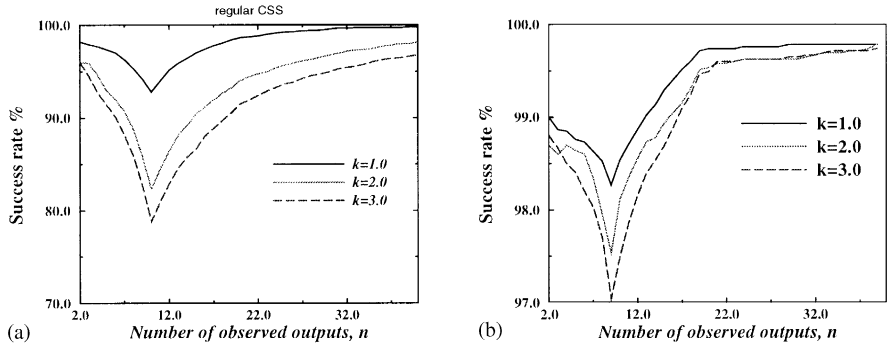

Fig. 7. Identifying transformed versions of the input query, k is the shear ratio and represents the measure of deformation (see Section 6). (a) CSS with arc length. (b) CSS with affine length.

success rate for the particular type of the CSS representation and for different values of n, the number of observed outputs.

Starting from Fig. 7(a), we observe that the conventional CSS image shows good results. For example, with $k = 1$ and in spite of severe deformation, more than 93% of outputs are always the affine transformed versions of the input query. This figure drops to 80% as k increases but it is still reasonably large.

With affine length parameterisation, the method shows much better results. Almost all affine transformed versions of an input query appear among the first outputs of the system. The results are also robust with respect to k, the shear ratio.

In conclusion, we observe the following.

Regular CSS image is quite robust with respect to affine transforms. Since the transformation is applied mathematically, the effects of pre-processing noise has not been considered. In real world applications [30], when the object boundaries must be extracted from images taken from different camera viewpoints, noise changes the object boundaries dramatically. However, we expect that using affine length instead of arc length improves the performance of the method even in presence of such noise.

6.1. Objective evaluation with classified database

Here we have chosen 76 shapes and classified them in 10 different groups as presented in Fig. 8. We produced 9 transformed shapes from each original shape. As a result, a group with 8 members had 80 members after adding the transformed shapes. The whole database then had 760 shapes in 10 different groups.

In order to assign a performance measure to the method using this classified database, we chose every member of each group as the input query and asked the system to find the best n similar shapes to the input from the database. We then observed the number of outputs, m, which are from the same group as the input. The *success rate* of the system is described by Eq. (7), where n is the number of observed outputs and m_{max} is the maximum possible value of m. Note that m_{max} is equal to n if n is less than the number of group members; if not, m_{max} is equal to the number of group members. The success rate of the system for the whole database is the average of the success rates for each input query. We chose different values for n and for each case, computed the success rate of the methods. The results are presented in Fig. 9(a) and (b). We observe that both arc length and affine length parameterisations lead to good results. However, the results for CSS with affine length are better and are not sensitive to the measure of affine deformation.

7. Comparison with other methods

Fourier descriptors [31] and moment invariants [32,33] have been widely used as shape descriptors in similarity transform environments. Both methods represent the global appearance of a shape in their most important components. For example, the largest magnitude component of Fourier descriptors represents the dimensions of the best fitted ellipse of the shape. Since affine transformation changes the global appearance of a shape, it is expected that the performance of these methods is negatively affected under the transformation.

The modified versions of these methods have been introduced to deal with affine transformation [8–10]. They are used in object recognition application and clustering a small number of shapes. However, it has been shown that the improvements in modified versions are not very significant in comparison to the conventional versions. Considering the fact that the implementation of the modified versions is not a straight forward task, we decided to examine the conventional versions of these methods and compare the results with the results of our

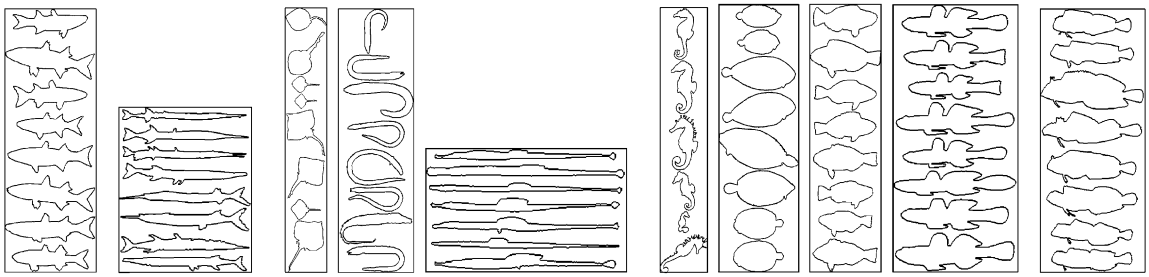

Fig. 8. Classified database used for objective evaluation consists of 10 different classes. Note that for each original shape, nine transformed versions are also generated. Therefore, the actual size of the database is 10 times larger than the size of this database.

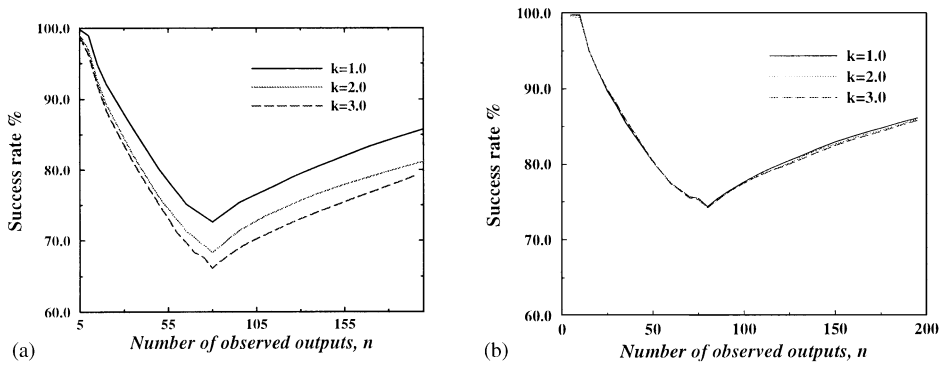

Fig. 9. The results of objective evaluation for different approaches. All terms are defined in Section 6.1. (a) CSS with arc length. (b) CSS with affine length.

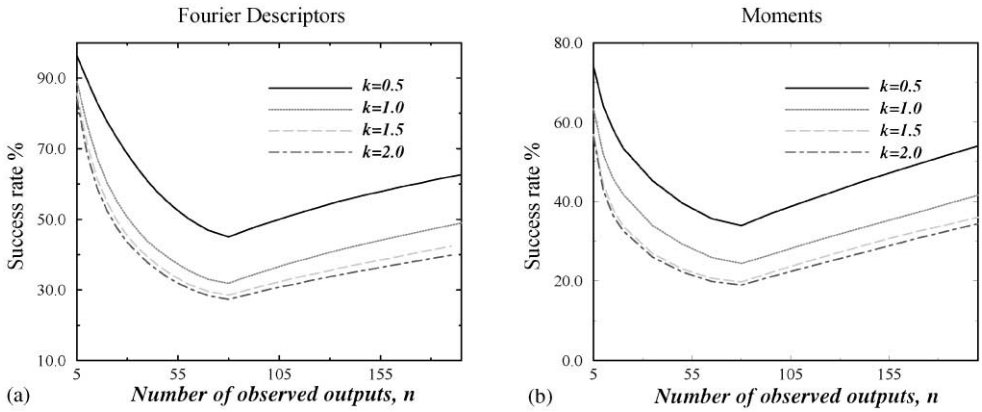

Fig. 10. The results of objective evaluation for different approaches. All terms are defined in Section 6.1.

method. We will observe that the difference between the performance measure of our method and the performance measure of each of these methods is very large. Even if we allow 10–15% improvement for the modified versions of the methods, their performance are still well behind the performance of the CSS representation.

We implemented the method described in Ref. [31] to represent a shape with its normalised Fourier descriptors. Every original shape and its transformed versions were represented by their first 20 components. The Euclidean distance was used to measure the similarity between the two representations.

The results for different values of k are presented in Fig. 10a. The minimum of the performance measure for different values of k is around 30%, compared to 70% of the CSS method. At the same time, the slope of the plots

after the minimum points is not sharp which means that most of the missing models are not ranked even among the first 200 outputs of the system.

For moment invariants, each object is represented by a 12 dimensional feature vector, including two sets of normalised moment invariants [32], one from object boundary and the other from solid silhouette. The Euclidean distance is used to measure the similarity between different shapes. The results are presented in Fig. 10b. Comparing this plot with the plots of Fig. 9 for the same shear ratio, a large difference between the performance of the CSS representation and this method is observed.

8. Concluding remarks

The maxima of curvature scale space (CSS) image have been used to represent closed planar curves in shape similarity retrieval under affine transforms. Two types of representations were examined. In conventional forms, arc length parametrisation is used to resample the curve at first stage of smoothing. We also examined the utility of using affine length instead of arc length to parametrise the curve prior to computing its CSS image. In different sections of this paper, we reviewed the background of the representations as well as parametrisations. We then carried out a number of experiments to compare the performance of our shape similarity system using these two approaches.

We constructed a database of 5000 contours using 500 real object boundaries of marine creatures and 4500 contours which are the affine transformed versions of real objects.

We observed that the performance of the CSS representation in shape similarity retrieval under affine transforms is promising and improves further by using affine length parametrisation. As a result, we believe that the method can also be used to represent and recognise 3-D objects using a small number of silhouette contours obtained from different view points [30]. Since the CSS representation is robust under affine transform, it can be used to represent each silhouette contour.

In conclusion, the CSS representation and its associated matching algorithm can be used for the purpose of shape similarity retrieval in an affine transformed environment and also for 3-D free-form object recognition. The CSS shape descriptor has been selected for MPEG-7 standardization.

Acknowledgements

Sadegh Abbasi is on leave from the University of Guilan, Rasht, Iran. He is grateful to the Ministry of Culture and Higher Education of Iran for its financial support during his research studies.

References

[1] S. Loncaric, Survey of shape analysis techniques, Pattern Recognition 31 (8) (1998) 983–1001.

[2] A. Del Bimbo, P. Pala, S. Santini, Image retrieval by elastic matching of shapes and image patterns, Proceedings of the 1996 International Conference on Multimedia Computing and Systems, Hiroshima, Japan, June 1996, IEEE, Computer Society Press, Los Alamitos, CA, USA, pp. 215–218.

[3] R. Mehrotra, J.E. Gary, Feature-based retrieval of similar shapes, In Proceedings of the Ninth International Conference on Data Engineering, Vienna, Austria, April 1993, IEEE, Computer Society Press, Los Alamitos, CA, USA, pp. 108–115.

[4] D. Mumford, The problem of robust shape descriptions, First International Conference on Computer Vision, London, England, June 1987, pp. 602–606.

[5] W. Niblack, R. Barber, W. Equitz, M.D. Flickner, E.H. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, G. Taubin, The qbic project; querying images by content using color texture and shape, SPIE 1908 (1993) 173–187.

[6] E. Saber, A.M. Tekalp, Image query-by-example using region-based shape matching, Proceedings of SPIE—The International Society for Optical Engineering, Vol. 2666, 1996, pp. 200–211.

[7] S. Sclaroff, A.P. Pentland, Modal matching for corresponding and recognition, IEEE Trans. Pattern Anal. Mach. Intell. 17 (6) (1995) 545–561.

[8] K. Arbter, W.E. Snyder, H. Burkhardt, G. Hirzinger, Applications of affine-invariant Fourier descriptors to recognition of 3-D objects, IEEE Trans. Pattern Anal. Mach. Intell. 12 (7) (1990) 640–646.

[9] J. Flusser, T. Suk, Pattern recognition by affine moment invariants, Pattern Recognition 26 (1) (1993) 167–174.

[10] A. Zhao, J. Chen, Affine curve moment invariants for shape recognition, Pattern Recognition 30 (6) (1997) 895–901.

[11] Z. Huang, F.S. Cohen, Affine-invariant B-spline moment for curve matching. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1994, pp. 490–495.

[12] H.W. Guggenheimer, Differential Geometry, McGraw-Hill, New York, 1963.

[13] D. Cyganski, R.F. Vaz, A linear signal decomposition approach to affine invariant contour identification, Proceedings of SPIE—Intelligent Robots and Computer Vision X: Algorithms and Techniques, Vol. 1607, 1991, pp. 98–109.

[14] G. Sapiro, A. Tannenbaum, Affine invariant scale space, Int. J. Comput. Vision 11 (1) (1993) 25–44.

[15] L. Alvarez, F. Guichard, P.L. Lions, J.M. Morel, Axioms and fundamental equations in image processing, Arch. Rat. Mech. 12 (1993) 199–257.

[16] D. Cyganski, T.A. Cott, J.A. Orr, R.J. Dodson, Development, implementation, testing, and application of an affine transform invariant curvature function, Proceedings of the First International Conference on Computer Vision, London, England, June 1987, pp. 465–500.

[17] B.B. Kimia, K. Siddiqi, Geometric heat equation and nonlinear diffusion of shapes and images, Comput. Vision Image Understanding 64 (3) (1996) 305–332.

[18] K. Siddiqi, B.B. Kimia, A. Tannenbaum, S.W. Zucker, Shapes, shocks and wiggles, Image Vision Comput. 17 (5) (1999) 365–373.
[19] B. Kimia, A. Tannenbaum, S.W. Zucker, Toward a computational theory of shape: an overview, Proceedings of the First European Conference on Computer Vision, Antibes, France, April 1990, Springer, Berlin, pp. 402–407.
[20] T. Cohignac, C. Lopez, J.M. Morel, Integral and local affine invariant parameter and application to shape recognition, Proceedings of the International Conference on Pattern Recognition (ICPR), Vol. 1, 1994, pp. 164–168.
[21] F. Mokhtarian, S. Abbasi, J. Kittler, Efficient and robust retrieval by shape content through curvature scale space, Proceedings of the First International Workshop on Image Database and Multimedia Search, Amsterdam, The Netherlands, August 1996, pp. 35–42.
[22] F. Mokhtarian, S. Abbasi, J. Kittler, Robust and efficient shape indexing through curvature scale space, Proceedings of the sixth British Machine Vision Conference, BMVC'96, Vol. 1, Edinburgh, September 1996, pp. 53–62.
[23] F. Mokhtarian, A.K. Mackworth, Scale-based description and recognition of planar curves and two-dimensional shapes, IEEE Trans. Pattern Anal. Mach. Intell. 8 (1) (1986) 34–43.
[24] L. Alvarez, P.L. Lions, J.M. Morel, Image selective smoothing and edge detection by nonlinear diffusion. II, SIAM J. Numer. Anal. 29 (1992) 845–866.
[25] F. Mokhtarian, A.K. Mackworth, A theory of multi-scale, curvature-based shape representation for planar curves, IEEE Trans. Pattern Anal. Mach. Intell. PAMI-14 (1992) 789–805.
[26] F. Mokhtarian, Silhouette-based isolated object recognition through curvature scale space, IEEE Trans. Pattern Anal. Mach. Intell. 17 (5) (1995) 539–544.
[27] S. Abbasi, F. Mokhtarian, J. Kittler, Reliable classification of chrysanthemum leaves through curvature scale space. Proceedings of the Scale-Space'97 Conference, Utrecht, Netherlands, July 1997, pp. 284–295.
[28] S. Abbasi, Curvature scale space in shape similarity retreival, Ph.D. thesis, Centre for Vision, Speech and Signal Processing, University of Surrey, Guildford, GU2 5XH, England, 1998.
[29] P.H. Winston, Artificial Intelligence, Addison-Wesely, Reading, MA, 1979.
[30] S. Abbasi, F. Mokhtarian, Shape similarity retrieval under affine transform: Application to multi-view object representation and recognition, Proceedings of the seventh IEEE International Conference on Computer Vision, ICCV99, Kerkyra, Greece, September, 1999, pp. 450–455.
[31] T.P. Wallace, P. Wintz, An efficient three-dimensional aircraft recognition algorithm using normalised fourier descriptors, Comput. Graphics Image Process. 13 (1980) 99–126.
[32] S. Dusani, K. Breeding, R.B. Mcghee, Aircraft identification by moment invariants, IEEE Transact. Comput. C-26 (1977) 39–45.
[33] M.K. Hu, Visual pattern recognition by moments invariants, IRE Trans. Inform. Theory IT-8 (1962) 179–187.

About the Author—FARZIN MOKHTARIAN is currently a Senior Lecturer at the Centre for Vision, Speech and Signal Processing at the University of Surrey in Guildford, England. He received his M.Sc. and Ph.D. in Computer Vision from the University of British Columbia, Vancouver, Canada and spent two years at NTT Basic Research Labs in Tokyo as a Research Scientist. He recently served on the Conference board and program Committee of the International Conference on Scale Space Theory in Computer Vision. His research interests include shape representation, object recognition, multi-scale shape analysis, and image database retrieval by shape content.

About the Author—SADEGH ABBASI is currently a research fellow at the Centre for Vision, Speech and Signal Processing at the University of Surrey. He received his B.Sc. and M.Sc. in Telecommunication Systems from the University of Tehran, Iran and worked for two years at Iran Telecommunication Research Centre. He then worked as a Lecturer at Shahid Chamran University in Ahvaz and Gilan University in Rasht, Iran. From October 1994, he joined the CVSSP where he received his Ph.D. in Computer Vision. His research interests include image database systems, shape similarity retrieval and pattern recognition.