2D Affine-Invariant Contour Matching Using B-Spline Model

Yue Wang, *Member*, *IEEE*, and
Eam Khwang Teoh, *Member*, *IEEE*

Abstract—This paper presents a new affine-invariant matching algorithm based on B-Spline modeling, which solves the problem of the nonuniqueness of B-Spline in curve matching. This method first smoothes the B-Spline curve by increasing the degree of the curve. It is followed by a reduction of the curve degree using the Least Square Error (LSE) approach to construct the Curvature Scale Space (CSS) image. CSS matching is then carried out. Our method combines the advantages of B-Spline that are continuous curve representation and the robustness of CSS matching with respect to noise and affine transformation. It avoids the need for other matching algorithms that have to use the resampled points on the curve. Thus, the curve matching error is reduced. The proposed algorithm has been tested by matching similar shapes from a prototype database. The experimental results showed the robustness and accuracy of the proposed method in B-Spline curve matching.

Index Terms—Curve matching, B-Spline model, curvature scale space, curve smoothing.

1 INTRODUCTION

OBJECT matching is the ultimate purpose of many image processing and computer vision applications. Since an object may undergo various affine transformations, the matching methods should possess the capability of handling such cases. A number of shape representations have been proposed for matching under an affine transformation. These include the affine invariant Fourier descriptors (FDs) [6] and moment invariants [9], [10]. In both methods, the emphasis is on using a parameterization that is robust with respect to affine transformation. The affine length is used to replace the arc length. However, the performance of such methods degrades in the presence of local noise and distortion. Moreover, in those methods that are based on moment [9], [10], the affine transform must be estimated. Affine invariant scale space is introduced in [11] as a curve evolution method and the affine curvature is proposed as well. However, this is not an explicit shape representation. The Curvature Scale Space (CSS) image is introduced in [12] as a shape representation for digital planar curves. This representation was computed by convolving the curve with a Gaussian function at different levels of scale and extracting the locations of the resulting curves' inflection points for matching. Recently, affine length and affine curvature have been used [14] in affine transformed environments. However, all the methods mentioned above suffer from inaccuracy and inefficiency due to higher order derivatives for affine length and affine curvature computation on the digital contour.

The B-Spline curve is one of the most efficient curve representations and possesses very attractive properties, such as compactness and continuity. However, very little work has been done on the use of the B-Spline for matching purposes. In [7], curve matching was achieved by using a similarity measure based on the reconstructed knot points of the B-Spline. As an extension of [7], a class of weighted B-Spline curve moments [9] was used to estimate the B-Spline control points. However, these methods cannot handle affine transformation, as arc length has been used. A residual error-based method that compares the sum of the residual error of the sampled points between two curves is suggested in [8]. In this method, if the sampled points lack good correspondences, the estimated affine transformation is not reliable. The algorithm in [1] uses the inflection points of the B-Spline curve and the sequence of area patches bounded by the contour and the line connecting two consecutive inflection points for contour matching. In this case, the noise effects or having few inflection points on the contours would degrade this algorithm. On the other hand, Gu and Tjahjadi [2] assume the arc-length to be affine-invariant and extract the significant corners to form the affine-invariant object features in their algorithm. The problems encountered in current B-Spline curve matching methods can be summarized as follows:

1. Research work so far tends not to employ the advantage of the continuous curve description of the B-Spline, but instead uses the discrete points sampled from the B-Spline curve in the matching process.
2. These algorithms are not affine invariant. In order to estimate the minimum error, the corner points or the discrete points resampled from a B-Spline curve with a certain arc-length are used. These methods may fail to handle affine transformation.
3. These algorithms are sensitive to noise and introduce sampling error. Error minimum estimations for curve matching are based directly on the sampled points, not the entire curve.

The problems mentioned above are mainly caused by the nonunique property of the B-Spline curve. Therefore, a shape descriptor based on the B-Spline curve should be sought.

Compared with existing methods, our method presented in this paper has three main novelties:

1. It relies on the whole, continuous B-Spline curve rather than the discrete resampled points used in almost all other methods, which tend to introduce error and lead to imprecise results.
2. The advantage of the B-Spline curve, which is continuous curve representation, is applied to calculate the affine curvature and affine length of the curve precisely and to generate the CSS image.
3. The proposed method can be used for affine invariant matching without estimating the affine transformation.

2 ALGORITHM FOR B-SPLINE CURVE MATCHING

This paper focuses on curve matching using the B-Spline curve, where the object contour is assumed to have been approximated and presented by the B-Spline. Details on how to approximate the object contour are beyond the scope of this paper; interested readers may refer to [3], [4].

Our method consists of two steps: 1) smooth the B-Spline curve and construct the CSS image and 2) extract the maximum of the CSS image and perform the matching [5]. In the first step, the B-Spline curve is smoothed by increasing the degree of the curve and subsequently reducing the degree to that of the original B-Spline curve using the Least Square Error (LSE) approach. In every smoothing procedure, the inflection points of the smoothed B-Spline are searched to form the CSS image until all of the inflection points have been smoothed out. In the second step, the maxima of the CSS

• *The authors are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639798. E-mail: {s2633175g, eekteoh}@ntu.edu.sg.*

Manuscript received 11 Dec. 2005; revised 22 Aug. 2006; accepted 23 Feb. 2007; published online 21 Mar. 2007.
Recommended for acceptance by L. Van Gool.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0696-1205.
Digital Object Identifier no. 10.1109/TPAMI.2007.1135.

image are extracted and matching is performed by comparing them with the same descriptors of the database objects.

2.1 B-Spline Curve Smoothing

The B-Spline curve is smoothed first by increasing the degree of the curve. In order to reduce the number of computational complications and to speed up the processing time, the LSE method is used to approximate the smoothed B-Spline curve with the same degree as the original B-Spline curve. As it is not defined directly on a set of discrete points on the curve, it avoids the problem caused by discretization. Moreover, as the smoothed B-Spline curve is represented by the same degree of the B-Spline, it allows for the straightforward computation of the geometric properties of the smoothed curve, such as affine length and curvature. Therefore, it is particularly suitable for forming the CSS image.

Curve Smoothing by Increasing the Degree of the B-Spline Curve. It is well known that, for the same set of control points, increasing the degree of the B-Spline curve representation will result in a smoother curve and converge to an ellipse. Moreover, the B-Spline representations of two sets of control points related by an affine transformation are related by the same affine motion, even in the case of increasing the degree of the B-Spline curve.

Degree Reduction for the Smoothed B-Spline Curve. For a B-Spline $\mathbf{g}_i(s)$ of degree k, assuming its control points are $\{Q_0, Q_1, \ldots, Q_n\}$, we want to approximate it by a B-Spline $\mathbf{h}_i(s)$ of degree $k-1$ with the control points $\{D_0, D_1, \ldots, D_n\}$. This is achieved by the LSE approach algorithm. The objective is to minimize

$$f = \sum_{i=0}^{n} \int_0^1 |\mathbf{g}_i(s) - \mathbf{h}_i(s)|^2 ds. \tag{1}$$

In order to get the control point $\{D_0, D_1, \ldots, D_n\}$ of B-Spline $\mathbf{h}_i(s)$, we should get the partial derivatives of $\{D_0, D_1, \ldots, D_n\}$ from (1), that is

$$\frac{\partial f}{\partial D} = \sum_{j=0}^{n} \sum_{i=0}^{n} \int_0^1 \frac{\partial}{\partial D_j} |\mathbf{g}_i(s) - \mathbf{h}_i(s)|^2 ds = 0. \tag{2}$$

By solving (2), we can obtain the control points $\{D_0, D_1, \ldots, D_n\}$ of the lower degree B-Spline. Here, we consider the case of approaching the B-Spline curve with a degree 5 by the B-Spline of degree 4. First, we assume, for a closed B-Spline with a degree 5, we have

$$\mathbf{g}_i(s) = \begin{bmatrix} s^4 & s^3 & s^2 & s & 1 \end{bmatrix} M_5 \begin{bmatrix} Q_i \\ Q_{(i+1)\bmod(n+1)} \\ Q_{(i+2)\bmod(n+1)} \\ Q_{(i+3)\bmod(n+1)} \\ Q_{(i+4)\bmod(n+1)} \end{bmatrix}, i = 0, 1, 2, \ldots, n, \tag{3}$$

where

$$M_5 = \frac{1}{24} \begin{bmatrix} 1 & -4 & 6 & -4 & 1 \\ -4 & 12 & -12 & 4 & 0 \\ 6 & -6 & -6 & 6 & 0 \\ -4 & -12 & 12 & 4 & 0 \\ 1 & 11 & 11 & 1 & 0 \end{bmatrix}. \tag{4}$$

For the B-Spline with degree 4, we have

$$\mathbf{h}_i(s) = \begin{bmatrix} s^3 & s^2 & s & 1 \end{bmatrix} M_4 \begin{bmatrix} D_i \\ D_{(i+1)\bmod(n+1)} \\ D_{(i+2)\bmod(n+1)} \\ D_{(i+2)\bmod(n+1)} \end{bmatrix}, i = 0, 1, 2, \ldots, n, \tag{5}$$

where

$$M_4 = \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}. \tag{6}$$

Furthermore, (2) can be converted to become (7).

$$B \cdot Q = A \cdot D, \tag{7}$$

where

$$Q = [Q_0 \; Q_1 \; Q_2 \; \ldots \; Q_n]^T, \tag{8}$$

$$D = [D_0 \; D_1 \; D_2 \; \ldots \; D_n]^T, \tag{9}$$

and

$$B = \begin{bmatrix}
\frac{15619}{20160} & \frac{15619}{20160} & \frac{477}{2249} & \frac{247}{20160} & \frac{1}{20160} & 0 & 0 & \cdots & \cdots & 0 & 0 & \frac{1}{20160} & \frac{247}{20160} & \frac{477}{2249} \\
\frac{477}{2249} & \frac{15619}{20160} & \frac{15619}{20160} & \frac{477}{2249} & \frac{247}{20160} & \frac{1}{20160} & 0 & 0 & \cdots & \cdots & 0 & 0 & \frac{1}{20160} & \frac{247}{20160} \\
\frac{247}{20160} & \frac{477}{2249} & \frac{15619}{20160} & \frac{15619}{20160} & \frac{477}{2249} & \frac{247}{20160} & \frac{1}{20160} & 0 & 0 & \cdots & \cdots & 0 & 0 & \frac{1}{20160} \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\cdots & 0 & \frac{1}{20160} & \frac{247}{20160} & \frac{477}{2249} & \frac{15619}{20160} & \frac{15619}{20160} & \frac{477}{2249} & \frac{247}{20160} & \frac{1}{20160} & 0 & 0 & \cdots & \cdots \\
\cdots & 0 & 0 & \frac{1}{20160} & \frac{247}{20160} & \frac{477}{2249} & \frac{15619}{20160} & \frac{15619}{20160} & \frac{477}{2249} & \frac{247}{20160} & \frac{1}{20160} & 0 & 0 & \cdots \\
\cdots & \cdots & 0 & 0 & \frac{1}{20160} & \frac{247}{20160} & \frac{477}{2249} & \frac{15619}{20160} & \frac{15619}{20160} & \frac{477}{2249} & \frac{247}{20160} & \frac{1}{20160} & 0 & \cdots \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\frac{247}{20160} & \frac{1}{20160} & 0 & 0 & \cdots & \cdots & 0 & 0 & \frac{1}{20160} & \frac{247}{20160} & \frac{477}{2249} & \frac{15619}{20160} & \frac{15619}{20160} & \frac{477}{2249} \\
\frac{477}{2249} & \frac{247}{20160} & \frac{1}{20160} & 0 & 0 & \cdots & \cdots & 0 & 0 & \frac{1}{20160} & \frac{247}{20160} & \frac{477}{2249} & \frac{15619}{20160} & \frac{15619}{20160} \\
\frac{15619}{20160} & \frac{477}{2249} & \frac{247}{20160} & \frac{1}{20160} & 0 & 0 & \cdots & \cdots & 0 & 0 & \frac{1}{20160} & \frac{247}{20160} & \frac{477}{2249} & \frac{15619}{20160}
\end{bmatrix}$$

$$A = \begin{bmatrix}
\frac{302}{315} & \frac{397}{840} & \frac{1}{21} & \frac{1}{2520} & 0 & 0 & \cdots & \cdots & 0 & 0 & \frac{1}{2520} & \frac{1}{21} & \frac{397}{840} \\
\frac{397}{840} & \frac{302}{315} & \frac{397}{840} & \frac{1}{21} & \frac{1}{2520} & 0 & 0 & \cdots & \cdots & 0 & 0 & \frac{1}{2520} & \frac{1}{21} \\
\frac{1}{21} & \frac{397}{840} & \frac{302}{315} & \frac{397}{840} & \frac{1}{21} & \frac{1}{2520} & 0 & 0 & \cdots & \cdots & 0 & 0 & \frac{1}{2520} \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\cdots & 0 & \frac{1}{2520} & \frac{1}{21} & \frac{397}{840} & \frac{302}{315} & \frac{397}{840} & \frac{1}{21} & \frac{1}{2520} & 0 & 0 & \cdots & \cdots \\
\cdots & 0 & 0 & \frac{1}{2520} & \frac{1}{21} & \frac{397}{840} & \frac{302}{315} & \frac{397}{840} & \frac{1}{21} & \frac{1}{2520} & 0 & 0 & \cdots \\
\cdots & \cdots & 0 & 0 & \frac{1}{2520} & \frac{1}{21} & \frac{397}{840} & \frac{302}{315} & \frac{397}{840} & \frac{1}{21} & \frac{1}{2520} & 0 & \cdots \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\frac{1}{2520} & 0 & 0 & \cdots & \cdots & 0 & 0 & \frac{1}{2520} & \frac{1}{21} & \frac{397}{840} & \frac{302}{315} & \frac{397}{840} & \frac{1}{21} \\
\frac{1}{21} & \frac{1}{2520} & 0 & 0 & \cdots & \cdots & 0 & 0 & \frac{1}{2520} & \frac{1}{21} & \frac{397}{840} & \frac{302}{315} & \frac{397}{840} \\
\frac{397}{840} & \frac{1}{21} & \frac{1}{2520} & 0 & 0 & \cdots & \cdots & 0 & 0 & \frac{1}{2520} & \frac{1}{21} & \frac{397}{840} & \frac{302}{315}
\end{bmatrix}.$$

Therefore, the control points D of the degree reduced B-Spline are:

$$D = A^{-1} B \cdot Q. \tag{10}$$

It should be pointed out that the LSE approach algorithm introduces some error into the approximation. However, the error is negligible. Experimental results can be seen in Fig. 1b. Fig. 1a presents the smoothed B-Spline curve by increasing degree ($k = 2^i, i = 2, 4, 6, 7, 8$) only. The same input shape is used in both figures. The shape of the B-Spline curve is smoothed very well by the proposed method and converges to an ellipse, as in Fig. 1a. The results of two smoothing methods, shown in Fig. 1, are virtually indistinguishable.

2.2 Construction of the CSS Image

To achieve an affine invariant parameterization, the normalized affine arc-length and affine curvature are used to construct the CSS image of a planar shaped curve. The normalized affine arc-length can be defined as

$$L(s') = \frac{\int_0^{s'} |\dot{\mathbf{g}}(s) \times \ddot{\mathbf{g}}(s)|^{\frac{1}{3}} ds}{\int_0^1 |\dot{\mathbf{g}}(s) \times \ddot{\mathbf{g}}(s)|^{\frac{1}{3}} ds}. \tag{11}$$

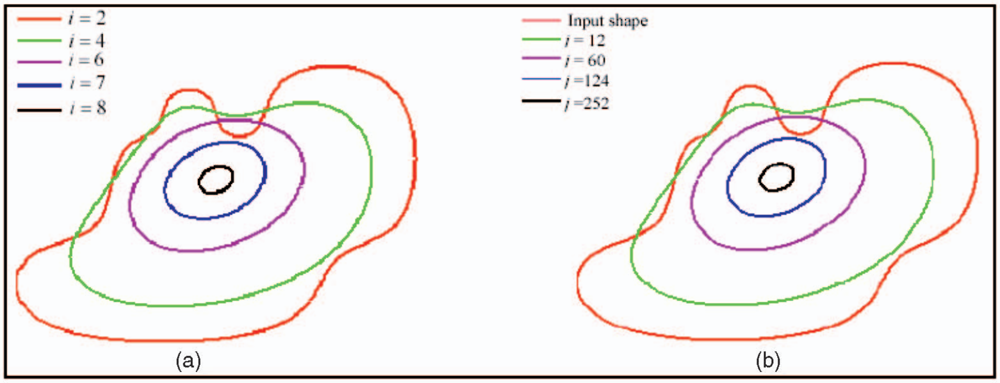

Fig. 1. B-Spline curve smoothing with increasing degrees and the proposed algorithm. (a) B-Spline curves with degree $k = 2^i$, $i = 2,$ 4, 6, 7, 8. (b) B-Spline curve smoothing based on the proposed method in Section 2.1. The red contour is the input shape (same as (a)) while the rest show the results with j times smoothing.

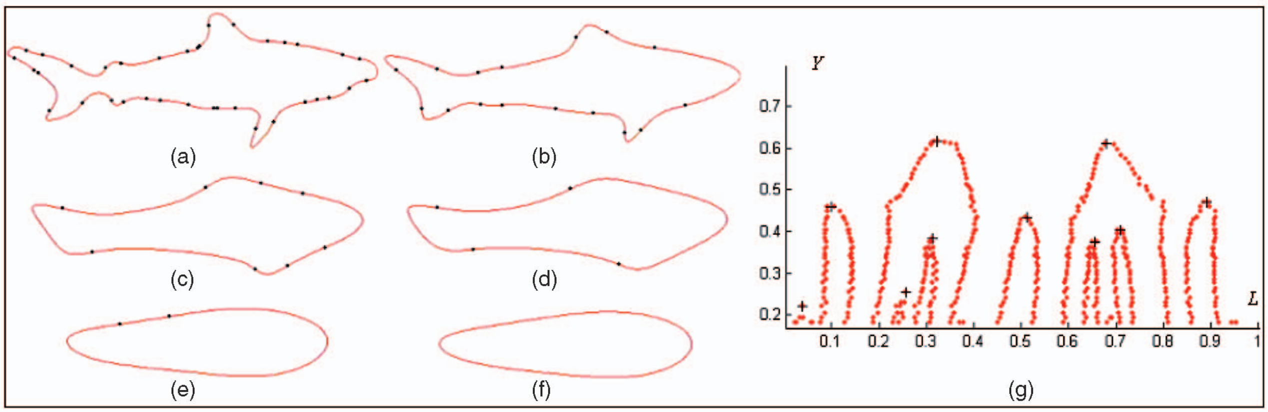

Fig. 2. CSS image construction for a fish shape. (a) Original image with 36 inflection points, (b) 16 inflection points, (c) 8 inflection points, (d) 4 inflection points, (e) 2 inflection points, and (f) 0 inflection points: The smoothing procedures for B-Spline curves using the proposed method. (g) Shows the constructed CSS image. Note that Y is defined in (13) and L indicates the locations of the inflection points on the normalized affine length of the whole smoothed curve.

For the affine curvature, we are only concerned with the locations of the inflection points (where the curvature is zero) and where we know the inflection point is affine invariant. For each curve segment i which is formed by the control points $\{Q_{i-1}, Q_i, Q_{i+1}, Q_{i+2}\}$ on a cubic B-Spline, the correspondence of parameter s' to an inflection point on the curve is given below:

$$s'_1 = \frac{-b + \sqrt[2]{b^2 - 4ac}}{2a}, \qquad s'_2 = \frac{-b - \sqrt[2]{b^2 - 4ac}}{2a}, \tag{12}$$

where

$$\begin{aligned} a &= (Q_{i-1} \times Q_i - 2Q_{i-1} \times Q_{i+1} + Q_{i-1} \times Q_{i+2} + 3Q_i \times Q_{i+1} - 2Q_i \\ &\quad \times Q_{i+2} + Q_{i+1} \times Q_{i+2}), \\ b &= (-3Q_{i-1} \times Q_i + 4Q_{i-1} \times Q_{i+1} - Q_{i-1} \times Q_{i+2} - 3Q_i \\ &\quad \times Q_{i+1} + Q_{i+1} \times Q_{i+2}), \\ c &= (2Q_{i-1} \times Q_i - 2Q_{i-1} \times Q_{i+1} + 2Q_i \times Q_{i+1}), \end{aligned}$$

s' has to be real and confined to [0, 1).

In a conventional CSS image construction, the vertical of the CSS image is defined by the width σ of a 1D Gaussian function. A new definition is required for the vertical axis of the CSS image when using our method. One possible way is to base it on the number of repeated degree reduction. However, we found that this may not be very accurate in some cases, especially with shapes containing a shallow concave. Although the original shape may not be "deep," the formed convex would be very "tall" in the CSS image. To solve this problem, 1-(the ratio of the affine arc-lengths of original shape and current smoothed shape) has been used as the definition of the vertical of CSS image

$$Y(m) = 1 - \frac{\sum_{i=0}^{n} \int_0^1 \left|\dot{\mathbf{g}}_i^m(s) \times \ddot{\mathbf{g}}_i^m(s)\right|^{\frac{1}{3}} ds}{L_{original}}, \tag{13}$$

where m is the times of the shape smoothing, $L_{original}$ is the affine arc-length of the original shape

$$L_{original} = \sum_{i=0}^{n} \int_0^1 |\dot{\mathbf{g}}_i(s) \times \ddot{\mathbf{g}}_i(s)|^{\frac{1}{3}} ds. \tag{14}$$

If we keep smoothing the object (no matter what the shape), the end result will shrink into its centroid. Therefore, the definition of (13) will also normalize the vertical of the CSS image to [0,1). If we determine the locations of the inflection points of every Y on the normalized affine length of the whole smoothed curve during the B-Spline smoothing procedure, the resulting points in the plane (L, Y) can be displayed, where L is the normalized affine length of current smoothed curve and is from 0 to 1. Y is an approximation defined in (13). One example of the construction of the CSS image, which follows the above definition, appears in Fig. 2.

2.3 Extraction of the Maxima of the CSS Contour

The locations of the CSS contour maxima are extracted as the shape descriptors for the CSS curve matching. Note that, usually, the small contours of the CSS image represent the noise or small ripples of the input curve. In order to simplify the computation and speed up the matching procedure, the small maxima are not included in the representation. We used a top-down search method in our experiments to extract the maxima of the convex in the CSS image.

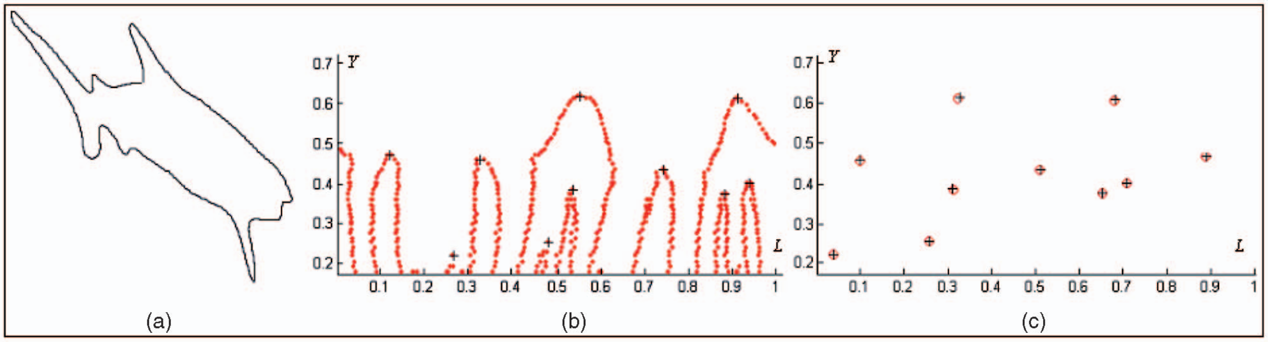

Fig. 3. CSS image for the affine transformed shape of Fig. 2a. (a) The affine transformed shape of Fig. 2a. (b) The constructed CSS image of (a). (c) A comparison of the maxima of the CSS images of the shapes. "O" denotes the maxima of the CSS image for the original shape, while "+" denotes the maxima of the CSS image of the affine transformed shape.

First, the higher points (the greater value of Y) are connected to the closest second higher points in the CSS image. Thus, the branches of the contours can be formed. Second, we locate the peaks of both branches of a contour in the CSS image and choose the midpoints of the line connecting the pair as the maxima of the CSS image. If a maximum is less than 0.3 of the largest maximum of the same CSS image, it will be removed from the matching process. As a result, only the significant concavities and convexities of a shape will contribute to the representation and matching processes. For instance, in the CSS image shown in Fig. 2g, there are 10 maxima.

2.4 Curvature Scale Space Matching

The extracted maxima of the models are sorted according to their vertical coordinates during the maxima extraction process. The complete matching algorithm mainly follows the conventional CSS matching algorithm [13]. It is briefly described as follows:

1. The largest scale maximum of the input shape is mapped to the largest scale maximum of the model. A CSS shift parameter d is computed for each maximum. This parameter is used to compensate for the effect of the different start points or a change in the orientation.
2. The largest scale image curve CSS maximum (which has not been mapped) is selected. Its shift parameter d is applied to a map that indicates the maximum for the model CSS image. The nearest model curve CSS maximum (which has not been mapped) is located. If the two maxima are at a reasonable horizontal distance (0.2 of the maximum possible distance) from one another, we define the matching value as the straight line distance between the two maxima. Otherwise, we define the height of the image curve CSS maximum as the matching value. If there are no more image curve CSS maxima left, we define the matching value as the height of the highest model curve CSS maximum that has not been matched. Likewise, if there are no more model curve CSS maxima left, we define the matching value as the height of the selected image curve maximum. We then record the matching value.
3. Steps 1 and 2 are repeated with the second largest scale maximum of the input shape to the largest scale maximum of the model, until all the major maxima have been tested.

In this case, the shape with the lower matching value will be used as the best match to the input shape. Here, the mirror image of the input shape should also be matched to the existing models of the database. Using the maxima of the input shape, we can easily calculate a new set of maxima, which belong to the CSS image of the mirror shape of the input. We can then repeat the matching algorithm for the new set and consider the lowest matching value of the two.

3 Experimental Results and Discussion

To evaluate our algorithm, two sets of the experimental results are presented in this section to demonstrate the performance of the affine invariant construction of the CSS image and object matching using the B-Spline model.

3.1 Affine Invariant Construction of the CSS Image

In our matching algorithm, the property of the affine invariant of the maxima of the CSS contour ensures affine invariant matching. Fig. 3a shows a shape C', which is related to the original shape C in Fig. 2a via an affine transformation.

In our experiment, the starting control point of C' was chosen randomly. The CSS image for the original shape and affine transformed shape are presented in Fig. 2g and Fig. 3b, respectively. In the first impression, these two CSS images are almost the same if we compensate for the horizontal shift. For an easy qualitative comparison, the overlaps of two sets of maxima (using the matching algorithm presented in Section 2.4) are shown in Fig. 3c. It can be observed that the maxima whose values of Y are lower than 0.6 are well matched. There are only two maxima that are greater than 0.6 in Y, which are not as well matched as the others. This is because the greater the value of Y, the more smoothing procedures have been performed for the B-Spline curve. As such, the likelihood of error increases, but it is still within the acceptable range. It confirms that the proposed CSS image construction can be used for handling the affine transformed shape and, thus, an affine invariant matching.

3.2 Object Curve Matching Using B-Spline Curve

A database that contains 1,100 marine creature shapes [15] has been converted to the B-Spline representation for this experiment. It demonstrates a great range of shape variations. The maxima of the CSS image for every shape have been extracted for the purpose of matching. The algorithm presented above is used to find shapes similar to the input shapes from this prototype database. The inputs use the shapes that already exist in the database. The first output of the matching procedure is always identical to the input shape with a zero matching value.

The example shown in Fig. 4a1 is an input contour, which is a shark. Similar contours that have been matched using the proposed algorithm are shown in Figs. 4a2, 4a3, 4a4, 4a5, 4a6, and 4a7. In this example, the scale is slightly different. The scale used in Fig. 4a3 is larger than the others. As our algorithm has the capability of handling the case of the mirror-image of the input, we

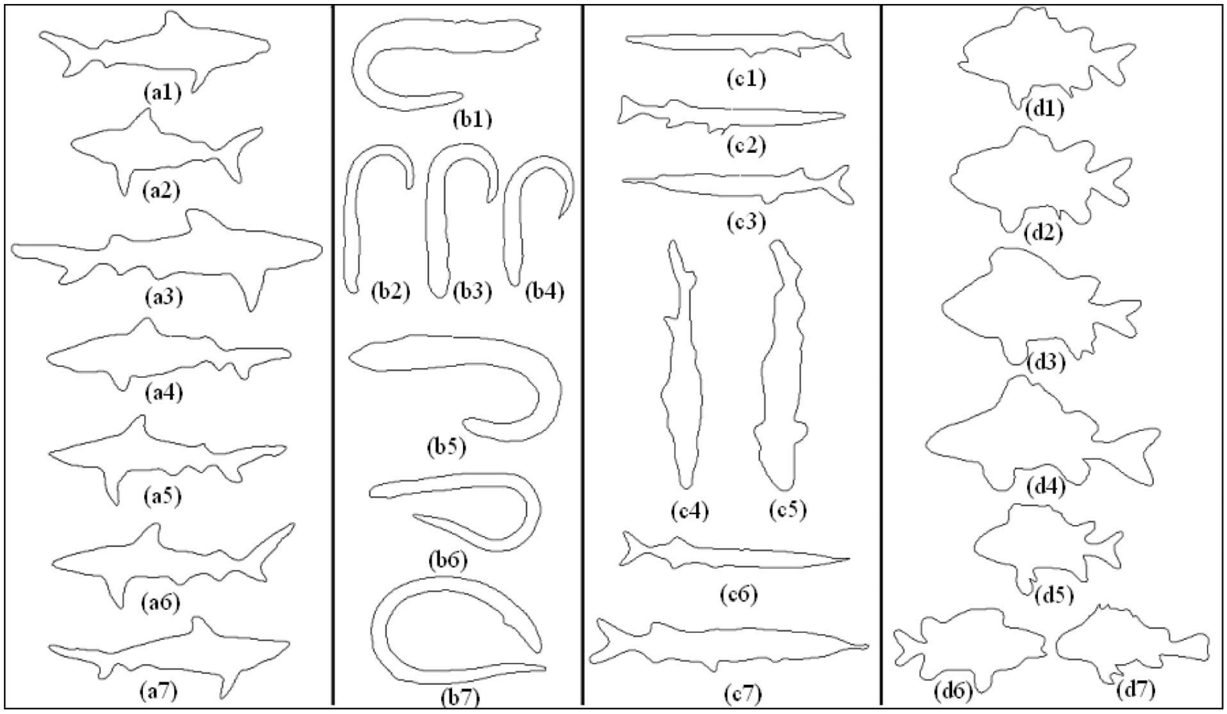

Fig. 4. Matching results. The first row is the contour to be matched, while the rest are the best matched contours 1 to 6 from the database, respectively. (a1)-(a7) Shark contour. (b1)-(b7) Eel contour. (c1)-(c7) Slim fish contour. (d1)-(d7) Fish contour.

observe that the matched results shown in Figs. 4a2, 4a4, 4a5, and 4a6 are the mirror-images of Fig. 4a1. All the matched shapes obtained using the proposed algorithm can be identified as that of a shark. The input shown in Fig. 4b1 is the contour of a sea eel. This is a shape that is curly in nature. The first six best matching results are shown in Figs. 4b2, 4b3, 4b4, 4b5, 4b6, and 4b7. Note that the matching results are different in scale. Figs. 4b5 and 4b6 illustrate the mirror-images of the input, while Figs. 4b2, 4b3, and 4b4 illustrate not just the mirror-images of the input, but also the rotations of the input image. As the rotation causes only a circular shift in the CSS image, it does not affect the performance of the matching results. The matching of a narrow type of fish (Fig. 4c1) is shown in Figs. 4c2, 4c3, 4c4, 4c5, 4c6, and 4c7. The input contour is quite slim. The matching algorithm successfully identifies similar contours in the database.

Figs. 4d2, 4d3, 4d4, 4d5, 4d6, and 4d7 show contour matching whose input indicates significant differences in the contour (Fig. 4d1). Compared to Fig. 4c1, it has more ripples. It can be observed that the matching results may not have the exact same ripples as the input, such as the mouth. The mouth of the input shape is open while similar cases of an open mouth exist only in Figs. 4d2 and 4d6. The matching results vary in size and Fig. 4d6 is the mirror-image of the input shape. However, the overall shapes of these seven objects are similar. The examples shown in Fig. 4 demonstrate that the proposed matching algorithm is robust with respect to the scale and orientation changes, as well as the mirror-image of the shape.

3.3 Comparison with the Affine Invariant Fourier Descriptors

A qualitative comparison between the proposed algorithm and the affine invariant FDs [6] is presented in this section.

1. The affine invariant FDs do not lose contour information compared to the method presented in this paper. This is because the LSE algorithm used in our method introduces an error in approximation.
2. Fourier-based features describe the global geometric properties of the object contours and local noise and distortion may degrade the object's performance in the calculation of a derivative with the discrete sampled points. The features used in our method only describe local properties, and they are robust against noise, as noise can only cause small ripples in the CSS image.
3. As there are fewer parameters in the shape descriptor extracted from the CSS image in our method compared to that in the FDs, the matching speed of the proposed algorithm is increased.
4. The processing time for forming the CSS image is hard to predict using our method, as it is not possible to know in advance when all the inflection points will be smoothed out for a given input curve. There is no such limitation in the affine invariant FDs.

4 Conclusion

This paper presented a two-step object matching algorithm using B-Spline as the curve representation. In the first step, the B-Spline curve was smoothed by increasing the degree of the curve, and it was approximated by using the LSE method to obtain the smoothed B-Spline curve with the same degree as the original B-Spline curve. In every smoothing process, the inflection points of the smoothed B-Spline were searched to form the CSS image. This process will only stop when all of the inflection points have been smoothed. In the second step, the maxima of the CSS contour were extracted as the feature vectors to represent the input, and matching was performed based on comparing them with the same descriptors of the database objects. The proposed method combined the advantages of B-Spline curves that are continuous curve representation and the robustness of the CSS matching with respect to noise and affine transformation. The B-Spline curve is particularly easy to use and accurate in calculating the curvature of the curve and generating the CSS image due to its continuous expression for a curve. It can be used for affine invariant curve matching without an estimation of affine transformation. The proposed method was tested for finding shapes that were similar to the shapes in a prototype of marine animals. The experimental results show the robustness and accuracy of the proposed algorithm. Our future work will extend the proposed method from 2D to 3D B-Spline curve matching.

REFERENCES

[1] W.S.I. Ali and F.S. Cohen, "Registering Coronal Histological 2-D Sections of a Rat Brain with Coronal Sections of a 3-D Brain Atlas Using Geometric Curve Invariants and B-Spline Representation," *IEEE Trans. Medical Imaging,* vol. 17, no. 6, 1998.

[2] Y. Gu and T. Tjahjadi, "Coarse-to-Fine Planar Object Identification Using Invariant Curve Features and B-Spline Modeling," *Pattern Recognition,* vol. 33, pp. 1411-1422, 2000.

[3] Y. Wang and E.K. Teoh, "Dynamic B-Snake Model for Complex Objects Segmentation," *Image and Vision Computing,* vol. 23, no. 12, pp. 1029-1040, 2005.

[4] Y. Wang and E.K. Teoh, "Object Contour Extraction Using Adaptive B-Snake Model," *J. Math. Imaging and Vision,* vol. 24, no. 3, pp. 295-306, 2006.

[5] Y. Wang and E.K. Teoh, "A Novel 2D Shape Matching Algorithm Based on B-Spline Modeling," *IEEE Proc. Int'l Conf. Image Processing,* pp. 409-412, 2004.

[6] K. Arbter, W.E. Snyder, H. Burkhardt, and G. Hirzinger, "Applications of Affine-Invariant Fourier Descriptors to Recognition of 3-D Objects," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 12, no. 7, pp. 640-646, July 1990.

[7] F.S. Cohen and J. Wang, "Modeling Image Curves Using Invariant 3D Object Curve Models—A Path to 3D Recognition and Shape Estimation from Image Contours," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 16, pp. 1-23, 1994.

[8] F.S. Cohen, Z. Huang, and Z. Yang, "Invariant Matching and Identification of Curves Using B-Splines Curve Representation," *IEEE Trans. Image Processing,* vol. 4, no. 1, pp. 1-10, Jan. 1995.

[9] Z. Huang and F.S. Cohen, "Affine-Invariant B-Spline Moments for Curve Matching," *IEEE Trans. Image Processing,* vol. 5, no. 10, pp. 1473-1480, Oct. 1996.

[10] A. Zhao and J. Chen, "Affine Curve Moment Invariants for Shape Recognition," *Pattern Recognition,* vol. 30, no. 6, pp. 895-901, 1997.

[11] G. Sapiro and A. Tannenbaum, "Affine Invariant Scale Space," *Int'l J. Computer Vision,* vol. 11, no. 1, pp. 25-44, 1993.

[12] F. Mokhtarian and A.K. Mackworth, "Scale-Based Description and Recognition of Planar Curves and Two-Dimensional Shapes," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 8, p. 3443, 1986.

[13] F. Mokhtarian, S. Abbasi, and J. Kittler, "Robust and Efficient Shape Indexing through Curvature Scale Space," *Proc. Seventh British Machine Vision Conf.,* vol. 1, pp. 53-62, 1996.

[14] F. Mokhtarian and S. Abbasi, "Affine Curvature Scale Space with Affine Length Parametrisation," *Pattern Analysis and Applications,* vol. 4, no. 1, pp. 1-8, 2001.

[15] http://www.ee.surrey.ac.uk/Research/VSSP/imagedb/demo.html, 2000.