Robust and Efficient Shape Indexing through Curvature Scale Space

Farzin Mokhtarian, Sadegh Abbasi and Josef Kittler
Vision Speech and Signal Processing Group
Department of Electronic & Electrical Engineering
University of Surrey
Guildford, Surrey GU2 5XH
England
F.Mokhtarian@ee.surrey.ac.uk

Abstract

The user of an image database often wishes to retrieve all images similar to the one (s)he already has. Using some features like texture, color and shape, we can associate a feature vector to every image in the database. A fast indexing method then can be used to retrieve similar images based on their associated vectors.

We use the maxima of curvature zero_crossing contours of Curvature Scale Space (CSS) image as a feature vector to represent the shapes of object boundary contours. The matching algorithm which compares two sets of maxima and assigns a matching value as a measure of similarity is presented in this paper. The method is robust with respect to noise, scale and orientation changes of objects. It is also capable to retrieve objects which are similar to the mirror-image of the input boundary.

We introduce the aspect ratio of the CSS image as a new parameter which can be used for indexing in conjunction with other parameters like eccentricity and circularity.

The method has been tested and evaluated on a prototype database of 450 images of marine animals with a vast variety of shapes with very good results. Since shape similarity is a subjective issue, in order to evaluate the method, we asked a number of volunteers to perform similarity retrieval based on shape on a randomly selected small database. We then compared the results of this experiment to the outputs of our system to the same queries and on the same database. The comparison indicated a promising performance of the system.

1 Introduction

Today, digital images have become the mainstream of information systems. The rapidly increasing number of images in many applications gives rise to the difficult problem of organizing them for the best and rapid access to their information content. When a user comes across a large number of images, he usually just knows something about the contents of his desired images and wishes to pick them up from the database. For example, a doctor may be interested in having all images of hearts with a certain kind of abnormality or a scientist in a geographic area may attempt to compile a set of satellite images containing Mississippi river.

There are two approaches to image databases indexing: *classical database* approach, and *computer vision* approach. In traditional approach the first and basic step in creating an image database system includes converting the image information into descriptive records and files which is usually performed manually or semi_automatically after preprocessing the images using conventional image processing techniques.[1]

Some visual properties of image such as texture and shape are difficult or nearly impossible to describe with text. Therefore, researchers in the computer vision camp have tried to describe these properties using some convenient feature vectors. Their attempts have been focused on the problem of *similarity retrieval* using these feature vectors. The objective is to allow users to specify an image, or a part of it, and the system should then find all images in the database *like* that (part of the) image.

In [4], the authors have used Polygonal approximation, while a set of features like boundary/perimeter, elongation (major axis/ minor axis), number of holes, etc, have been used in [3] for shape similarity retrieval. The authors in [7] have used a combination of heuristic shape features such as area, circularity, eccentricity, major axis orientation and a set of algebraic moment invariants. They have also used other features such as color, texture, and even sketch features.

1.1 Our approach

The main aim of our experiments was to examine the performance of a modified version of Curvature Scale Space (CSS) image matching [6] for comparing shapes of objects in an image database. We gathered a database of 450 colored images of marine animals, with every image containing one animal on a uniform background. We then computed the CSS image of every boundary and then found the maxima of CSS contours as a shape descriptor to compare images. The basic properties of the CSS representation which make it almost unique for shape similarity retrieval are as follows:

- It is robust with respect to noise, scale and orientation changes of the objects (See section 2.3 and figure 1).
- It retains the local information of the input shape. Every concavity or convexity on the shape has its own corresponding contour on the CSS image. Every point in horizontal axis of CSS image has its corresponding on the actual boundary.
- It is fast. In response to a query, the system computes the CSS image of the input boundary and extracts its maxima. The system then compares the extracted feature vector to those of some image candidates in the database. Both feature extraction and matching process are fast. Note that the speed criterion is vital in the case of large image databases. Also note that it is

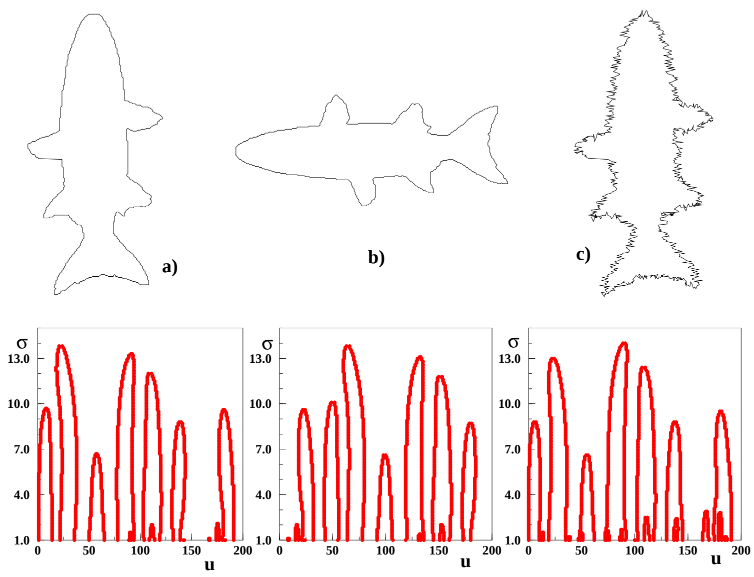

Figure 1: a) A boundary and its CSS image. b)Change in orientation causes a circular shift in CSS image. c) Noise creates small contours in CSS image.

possible to narrow down the search space by using some simple attributes like aspect ratio (section 3.1), eccentricity or compactness.

- It is reliable. The results of our experiments confirm the reliability of the representation and of the matching algorithm.
- Although we deal with simple images in our database, the representation and matching algorithm can be used without any change in the case of more complicated images provided that the boundaries of objects have already been extracted.

The following is the organisation of the remaining sections of this paper. In section 2 we explain the preprocessing of images. The method used to extract object boundaries, the method of computing curvature scale space image, and the algorithm for finding maxima of CSS image are described in this section. Section 3 explains the matching procedure and the experimental results are presented in section 4 .

2 Preprocessing

2.1 Image Segmentation

Our database consists of simple images of marine animals, with every image containing one animal in a uniform background. After the thresholding process, it is possible to track the external boundary of object to extract the object boundary contour. The resulting boundary may be noisy. However, since the locations of the maxima of the CSS image are robust with respect to noise, it does not cause any problems.

2.2 The Curvature Scale Space Representation

The curvature of a curve is defined as the derivative of the tangent angle to the curve. Consider a parametric vector equation for a curve:

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

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

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

If Γ is a closed planar curve, u can be the normalized arc length parameter which means:

$$\Gamma = \left\{ \, \big(x(u), y(u)\big) \; \big| \; u \in [0,1] \, \right\}$$

and the denominator in equation (1) will then be equal to one and we obtain:

$$\kappa(u) = \dot{x}(u)\ddot{y}(u) - \ddot{x}(u)\dot{y}(u) \quad .$$

In the rest of this paper we will assume that the curves are closed and planar and are expressed in terms of the normalized arc length.

There are several approaches in calculating the curvature of a digital curve [8]. We use the idea of *curve evolution* which basically studies shape properties while deforming in time. A certain kind of evolution can be achieved by Gaussian smoothing to compute curvature at varying levels of detail. If $g(u,\sigma)$ is a 1-D Gaussian kernel of width σ, then $X(u,\sigma)$ and $Y(u,\sigma)$ represent the components of *evolved* curve,

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

according to the properties of convolution, the derivatives of every component can be calculated easily :

$$X_u(u,\sigma) = x(u) * g_u(u,\sigma) \qquad\qquad X_{uu}(u,\sigma) = x(u) * g_{uu}(u,\sigma)$$

and we will have a similar formula for $Y_u(u,\sigma)$ and $Y_{uu}(u,\sigma)$. Since the exact forms of $g_u(u,\sigma)$ and $g_{uu}(u,\sigma)$ are known, the curvature of an evolved digital curve can be computed easily.

$$\kappa(u,\sigma) = \frac{X_u(u,\sigma)Y_{uu}(u,\sigma) - X_{uu}(u,\sigma)Y_u(u,\sigma)}{(X_u(u,\sigma)^2 + Y_u(u,\sigma)^2)^{3/2}} \tag{2}$$

As σ increases, the shape of Γ_σ changes. This process of generating ordered sequences of curves is referred to as the evolution of Γ.

2.3 Curvature Scale Space 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 database.

To normalize the arc length, we re-sample the boundary and represent it by 200 equally distant points. Therefore, the perimeter of all boundaries will be the same.

Considering the resampled curve as Γ, we can determine the locations of curvature zero crossing on Γ_σ, using the formula described in the previous paragraph. We start the process with $\sigma = 1$ and increase it by 0.1 at each level. 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 crossing (see figure 2).

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 the normalized arc length and σ is the width of the Gaussian kernel. The result of this process can be represented as a binary image called CSS image of the curve (see figure 1a). 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 figure 2 for $\sigma = 10$.

It should be clear why CSS image representation is robust with respect to scale, noise and change in orientation. A rotation of the object usually causes a circular shift of its representation which is easily determined during the matching process (compare figures 1a and 1b). Note that the effect of a change in the starting point is also the same. Due to the normalization, scaling does not change the representation, and as figure 1 shows, the noise may create some small contours on CSS image, but the main contours remain unaffected.

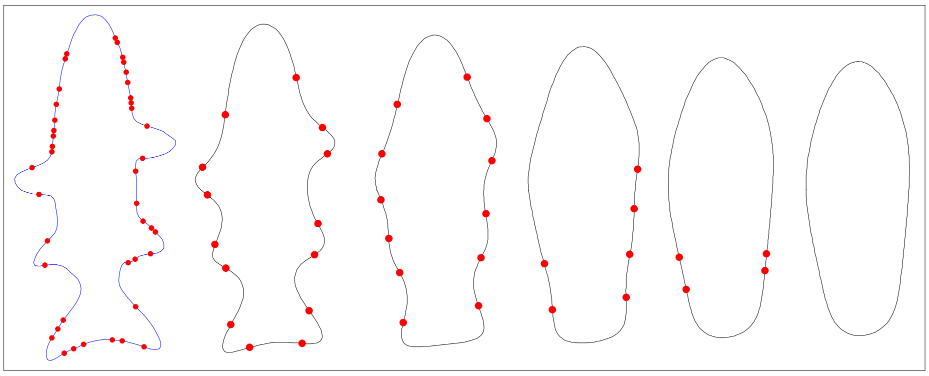

Figure 2: Shrinkage and smoothing of the curve and decreasing of the number of curvature zero crossings during the evolution, from left: $\sigma = 1, 4, 7, 10, 12, 14$.

2.4 Extracting maxima of Curvature Scale Space Contours

We represent every image in the database just with the locations of its CSS contour maxima. For example, the representation for the object in figure 1a will be as follows:

$$\{(22\ 13.8), (91\ 13.3), (109\ 12.0), (8\ 9.7), (181\ 9.6), (139\ 8.8), (57\ 6.7)\}$$

Note that usually the small contours on the CSS image represent some information about the existing noise on the actual object, so we just pick up those maxima which are higher than 0.2 the largest σ of the CSS image. The locations of the maxima are not readily available and must be extracted from the CSS image. A CSS contour is usually connected everywhere except in a neighborhood of its maxima. We find the peaks of both branches of a contour in the CSS image and consider the midpoint of the line segment joining the pair as a maximum of the CSS image.

3 Curvature Scale Space Matching

We assume that the user enters his query by sketching a boundary of his desired image or by pointing to an image. In each case, we do 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*. In this section we explain the algorithm of matching which is rather different from what is proposed in [6].

3.1 Indexing

Every CSS contour corresponds to a concavity or convexity in the actual image boundary. For example in figure 1a, there are seven concavities in the boundary of the object and there are seven contours and therefore seven maxima in the CSS image. In this figure, a concavity can be observed in the tail of the animal to which the third contour, smallest one, in the CSS is related . If the width of the CSS image, ie. the number of samples used to represent the image contour is the same, the longer and deeper a concavity , the taller the corresponding CSS contour. This means that if the image contour consists of long and/or deep concavities, the height of its CSS image is large. Therefore, by comparing just one integer, one can decide whether a model has any chance to be matched with the image or not.Note that if the width of CSS is not the same for all models, the parameter is simply replaced by *aspect ratio*, the ratio of height and width of every model, because if for example the number of samples (width of the CSS image) is doubled, every concavity becomes twice as long and it makes the height of the CSS image twice as high.

The aspect ratio is used for indexing, the first step of our algorithm is the comparison between aspect ratios of image and model. If the aspect ratio of model is within the range of 0.8 and 1.2 of aspect ratio of image, the matching algorithm will be applied, otherwise, the model is rejected.

3.2 The matching algorithm

After extracting the maxima of every model, we normalize their coordinates by dividing them to the number of samples, eg 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-coordinate (filter width) during the process of maxima extraction.

The matching algorithm which compares the two sets of maxima, one from the image and the other from the model is as follows.

1. Create a node consisting of the largest scale maximum of the image and the largest scale maximum of the model. If there are more than one maxima in the model which have a σ-coordinate close (within 80 percent) to the largest scale maximum of the image, create extra nodes consisting of the largest scale maximum of the image and that respective additional maximum of the model. Also create the same nodes for the second largest scale maximum of the image and the respective maxima of the model. Initialize the *cost* of each node to the absolute difference of σ-coordinates of the image and the model. Compute a CSS shift parameter α for each node :

   $$\alpha = U_m - U_i$$

   where U is the horizontal coordinate of a maximum, and i and m refer to image and model respectively. This parameter is used to compensate the effect of different start points or change in orientation.

2. Create two lists for each node obtained in step 1. The first list will contain the image curve maxima and the second list will contain the model curve maxima matched within that node at any point of the matching procedure. Initialize the first and second list of each node by the corresponding maxima determined in step 1.

3. Expand each node created in step 1 using the procedure described in step 4.

4. To expand a node, select the largest scale image curve CSS maximum (which is not in the first list) and apply that node's shift parameter computed in step 1 to map that maximum to the model CSS image. Locate the nearest model curve CSS maximum (which is not in the second list). If the two maxima are in a reasonable horizontal distance (0.2 of the maximum possible distance), define the cost of the match as the straight line distance between the two maxima. Otherwise, define the height of the image curve CSS maximum as the cost of the match. If there are no more image curve CSS maxima left, define the cost of match as the height of the highest model curve CSS maximum *not* in the node's second list. Likewise, if there are no more model curve CSS maxima left, define the cost of match as the height of the selected image curve maximum. Add the match cost to the node cost. Update the two lists associated with the node.

5. Select the lowest cost node. If there are no more model or image curve CSS maxima that remain unmatched within that node, then return that node as

the lowest cost node. Otherwise, go to step 4 and expand the lowest cost node.

6. Reverse the place of the image and the model and repeat steps 1 to 5 to find the lowest cost node in this case.

7. Consider the lowest node as the final matching cost between the image and the model.

Using this algorithm and considering its amendment which follows immediately in section 3.3, the system associates a matching value to every candidate and then displays the n best matched as its output.

3.3 The problem of mirror-images in the CSS matching

To represent a curve in terms of the normalized arc length, we need to consider a point on the curve as the origin. We also need to consider a direction on the curve as the increasing direction of the arc length.

It is possible to consider the same direction, say clockwise, for all boundaries of the database. In this case, if a model in the database is similar to the mirror-image of the input, the CSS image of the model will also be similar to the mirror-image of the CSS image of the input. Since by just circular shift it is not possible to map the corresponding maxima in this case, the above mentioned algorithm will fail to discover the similarity between the input and the model. The same problem will arise if increasing direction of the arc length is different on the input and the model boundaries.

Therefore, the mirror-image of the input should also be compared to the existing models of the database. Using the input maxima, we can easily calculate a new set of maxima which belongs to the CSS image of the mirror-image of the input. We can then either repeat steps 1 to 7 for the new set and consider the lowest matching cost between the two or construct new nodes in step 1 for the new set and expand all nodes simultaneously.

4 Results and discussion

We tested the proposed method on a database of 450 images of marine animals. Each image consisted of just one object on a uniform background. The system software was developed using the C language under Unix operating system. The response rate of the system was less than one second for every user query.

We discuss the experimental results by representing the response of the system to some queries. In the first example the input is an image which already exists in the database. The input image is shown in figure 3a and the output of the system is in figure 3c . The first output of the system is identical with the input image, with a zero match value. Note that the fourth and fifth output images are different just in scale. Also note the sixth output is similar to the mirror-image of the input.

In the second query, user has used a mouse to paint an outline of his desired object. Figure 3b shows the input which is quite noisy, and figure 3d consists of retrieved images. As this example shows, the system is also robust with respect to noise.

Other examples are presented in figures 3e and 3f, where the inputs and the first outputs of the system are identical and other outputs are similar to the inputs. These examples also show the variety of shapes of objects in our database.

The evaluation of the performance of the system is a difficult task, because shape similarity is a subjective matter. We selected 50 images from our prototype database randomly and created a small database. We then selected 20 inputs from this database and asked a number of volunteers to find the shapes similar to every input from the database. The results of the subjective test indicated that human judgements of shape similarity noticeably differ. Interestingly though, the ranking produced by our system always agreed quite closely with, at least, a subset of the human evaluators. The short lists of the top five shapes generated by the different judges always included the "closest" machine selected shape. These findings indicate that the proposed approach is promising.

5 Conclusions

This paper described a method to retrieve similar images from large image databases using their shape properties. A database of 450 images of marine animals was selected to test the method. The boundary of every object was extracted and the curvature scale space image of this boundary was computed. Then the maxima of this image were extracted to be considered as the shape descriptors. A fast and reliable algorithm for comparing these descriptors together with the results of several queries were presented.

Acknowledgements The authors are grateful to Dr. Radek Marik for his contribution in preprocessing of images. Sadegh Abbasi is grateful to the Ministry of Culture and Higher Education of Iran for its financial support during his research studies.

References

[1] Chang, S. A. and Hsu, A. " Image information systems, where do we go from here? " in IEEE Trans. on Knowledge and Data Engineering, VOL. 4, No 5, pages 431-442, October 1992.

[2] Eggleston, P. " Using constraints to incorporate domain knowledge " in Proc. of the Advances in Intelligent Robotic Systems, X, 1991.

[3] Eggleston, P. " Content based feature indexing and retrieval for image databases " in SPIE, Vol. 1819, 1992.

[4] Gary, E. and Mehrorta, R. " Shape similarity-based retrieval in image databases " in SPIE, Vol. 1662, 1992.

[5] Mokhtarian, F. and Mackworth, A. K. " A theory of multiscale, curvature-based shape representation for planar curves " in IEEE Trans. on Pattern Anal. Mach. Intell. , VOL. PAMI-14, No. 8, August 1992 pp789-805.

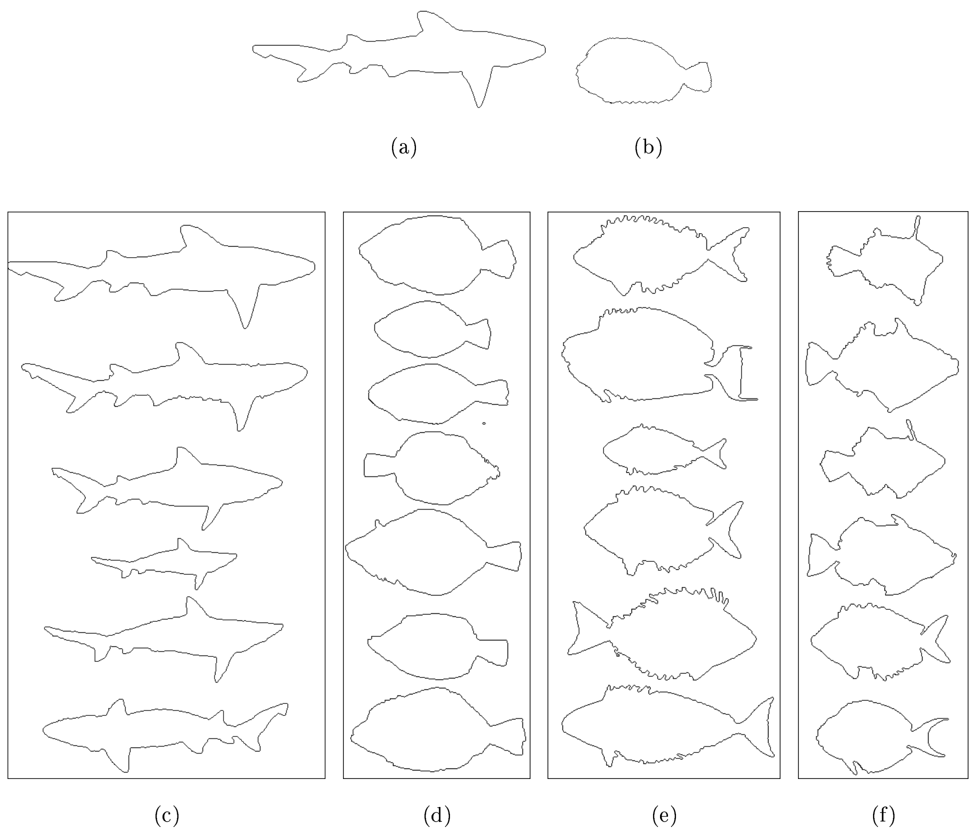

Figure 3: a)user points to an image. b) user paints a boundary. c to f) query results

[6] Mokhtarian, F. “ Silhouette-based isolated object recognition through curvature scale space ” in IEEE Transactions on Pattern Analysis and Machine Intelligence, May 1995, Vol.17, No.5, pp.539-544

[7] Niblack, W. *et al* , “ The QBIC project; querying images by content using color , texture and shape ” in SPIE, VOL. 1908, 1993.

[8] Tsai, D. and Chen M. “ Curve fitting approach for tangent angle and curvature measurement ” in Pattern Recognition, Vol 27, No 5, pp 699-711, 1994.