Conversion Database of the Shapes Into XML Data for Shape Matching

Noreddine GHERABI

Hassan [1]st University, FSTS
Department of Mathematics and
Computer Science, Labo LITEN

Mohamed BAHAJ

Hassan [1]st University, FSTS
Department of Mathematics and
Computer Science, Labo LITEN

Abstract— We present a new approach to the matching of 2D shapes using XML language and dynamic programming. Given a 2D shape, we extract its contour and which is represented by set of points. The contour is divided into curves using corner detection. After, each curve is described by local and global features; these features are coded in a string of symbols and stored in a XML file. Finally, using the dynamic programming, we find the optimal alignment between sequences of symbols. Results are presented and compared with existing methods using MATLAB for KIMIA-25 database and MPEG7 databases.

Keywords- XML; DOM; Shape descriptor; Shape matching; Dynamic Programming.

I. INTRODUCTION

Matching 2D shapes and measuring the similarity between shapes are important problems in Computer vision.

A large body of research has been devoted to shape matching, comparison and recognition. The most commonly used shape representation primitives are curves, point sets, and medial axes.

Many traditional curve matching approaches [1], [2], [3] use local invariant features (e.g., curvature) as descriptors.

Shapes have several properties that can be used for recognition and categorization, like shape, color, texture and brightness. Biederman [4] suggested that edge-based representations mediate object recognition. In his approach, color and texture of surfaces are used to define edges which are then used for recognition.

The goal of our work is to develop a descriptor based on the local and global information of the shape. These information are coded in string of symbols, these are compared using the Dynamic Programming approach. In [5, 6] dynamic programming is used to minimize a cost function that accounts for displacement of a contour in a pair of images from an image sequence. In [7] a DP approach has been used for shape matching and retrieval. The basic idea behind this approach is to represent each shape by a sequence of convex and concave segments using the inflection points extracted from the curvature and to allow the matching of merged sequences of small segments in a shape with larger segments in the other shape.

This paper looks into developing a shape descriptor for a contour of any shape and transforms it into string of symbols; which will be stored in an XML file. For each shape there is an XML file that corresponds to the features of the shape. Man Hing [8] uses this technique to extract the features information of the shape and represent this information in an XML format, this proposed system will use the XML (standard language) for querying different image databases.

Our approach aims to develop a simple and fast method of shape matching based on the transformation semantic data of the shape into XML format and compute the similarity between XML Files using Dynamic programming.

II. PROPOSED METHOD

Our approach to shape recognition is based on several steps summarized in Fig 1.

The first step is to analyze the contour of the shape to be studied. The contour is retrieved and represented by sets of points. After, the contour is divided into curves by using the technique of detection corner [11], this technique is detailed in section 2.1. Then each curve is transformed into a string of symbols. The string of symbols of each curve is

stored in a XML file (Section 2.2). Finally, we use the technique of dynamic programming for computing the similarity between the set of symbols stored in XML file (Section 2.3).
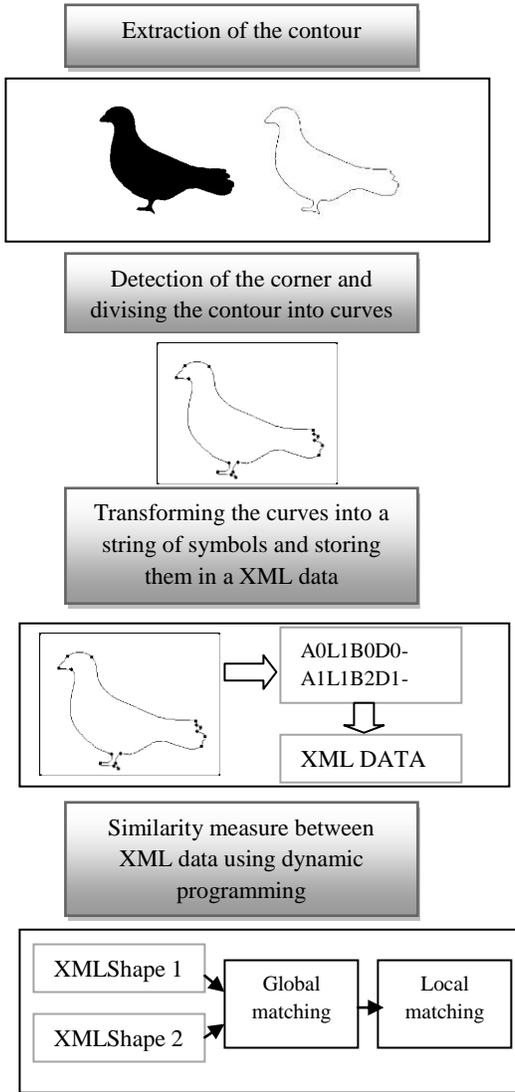

Fig.1: A block representation of the proposed algorithm for shape matching

A. Corner detector

Corners in images represent critical information in describing object features that are essential for pattern recognition and identification. There are many applications that rely on the successful detection of corners, including motion tracking, object recognition, and stereo matching [9,10,11]. As a result, a number of corner detection methods were proposed in the past.

In this paper, we use an algorithm developed by Xiao Chen and Nelson H. C. Yung [12], it works in two passes and defines a corner in a simple and intuitively appealing

way, as a location where a triangle of specified size and opening angle can be inscribed in a curve. The curve has to be generated previously using an edge detector. It is not required to be a closed curve. In the first pass the sequence of points is scanned and candidate corner points are selected. In each curve point p the detector tries to inscribe in the curve a variable triangle (p- , p, p+). The triangle varies between a minimum and a maximum square distance on the curve from p- to p, from p to p+ and the angle α ≤αmax (the value of αmax is defined) between the two lines a and b in Figure 2. Because the points are kept with their Cartesian coordinates we can easily compute α. Triangles are selected starting from point p outward and the number of admissible triangles is defined. At a neighborhood of points only one of these admissible triangles is selected. The one which has the smallest value for α. A value sharpness is assigned to p.

In the second pass the selection is refined and points that give the strongest response are marked as corners in the curve. This is done by selecting only points which have sharpness greater than that of their neighbors.
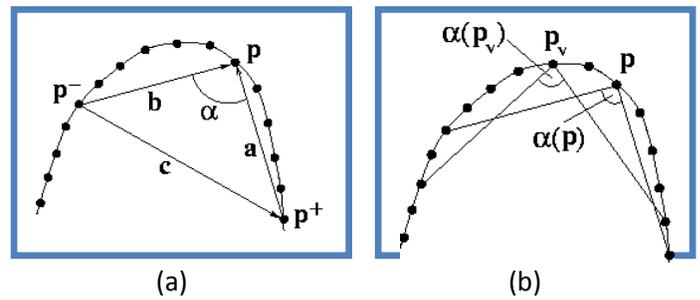

(a)   (b)

Fig 2: Detecting high curvature points.

(a) Determining if p is a candidate point.   (b) Testing p for sharpness

B. Symbolic representation and storing XML

1) Our approach for symbolic representation

The contour of the shape is retrieved and normalized to a set of points. This normalized contour used for feature extraction.

Our method uses local and global features to transform shape data into a new structure that supports measuring the similarity between shapes in an efficient manner, using the corner detection, the contour is segmented into a set of primitives (line, convex and concave curves) and described by the features: Ai, li, Dgi, βi, where :

- Ai is the area of the triangle enclosing the chord and the arc between the inflection points Pi and Pi+1, this area is calculated using Heron's formula (See Fig 2):

$$Area = \sqrt{s(s-a)(s-b)(s-c)} \qquad (1)$$

Where: $s = \dfrac{a+b+c}{2}$ is the Semiperimeter,
or half of the triangle's perimeter.

- l_i is the length of Curve (Ci,)

- Dgi is the Degree of concavity or convexity (Dgi=di/li) is computed as the ratio of the maximum of distances from points on the curve to associated chord and the distance of the chord of (Ci)

- β_i is the angle traversed by the tangent to the segment from inflection point Pi to inflection point Pi+1and shows how strongly a section is curved(See Fig. 3).
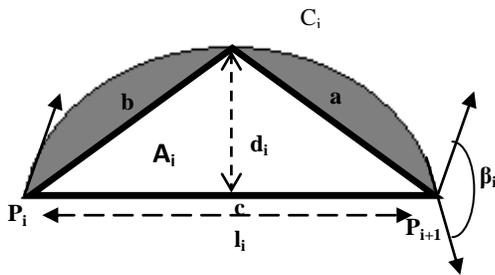

Fig.3: shape descriptor of the curve Ci

Now we will see how to transform the shape into a set of symbols.

The area Ai is computed and quantized in three bins (A0, A1, A2) corresponding to a zero, small and large area. The same, for each curve of the contour, the length of the segment [Pi, Pi +1] is computed, this normalized distance li is quantized in three bins (L1, L2, L3) corresponding to a small, medium and large distance of li. Next, the angle βi is computed and quantized in different five bins between [0, π] (B0,B1, B2, B3, B4), B0 for βi=0. Finally, the values Dgi of each curve is computed and quantized in three bins (D0, D1, D2) corresponding to zero, a small and large values of Dgi.

Fig.4: The mapping obtained by the algorithm of a given contour into a string of symbols.

Our algorithm converts the contour of the shape into sequences of symbols, for example the Mapping obtained by the algorithm of a curve (a) illustrated in figure 4 is : A2L2B2D2, A2 corresponding to a large area, L2 corresponding to a medium distance of Li, βi is quantized in the bin B2 and D2 corresponding to a large value of Dgi.

2) Writing in XML data

In this paper we propose XML language for describing the features of the shape structured in an order where each contour is associated to one descriptor written in a specific file XML. This technique for converting into XML was already used in our previous work [12].

To write the shape descriptor, each curve is represented by a set of parameters, these parameters are encoded using XML tags. to construct the XML structure we used the technique of DOM in Matlab. We use the syntax of XML to write our outline of the shape as follows:

The curve is defined by its type and described by the parameters (Ai, Li, Dgi, βi)

- Concave/ Convex/Right curve

< C number ='​'> // curve and its number in the outline

<Type > CC or CV or R <Type/> // Type of curve (concave/convex) or right line

    <Ai>   </Ai> area Ai

    < li >   length of curve

    <beta_i>   </ beta_i > angle βi

    < Dgi >   </Dgi > degree of gravity

</C>

An iterative process is presented to describe the shape using XML, this process is in the following algorithm:
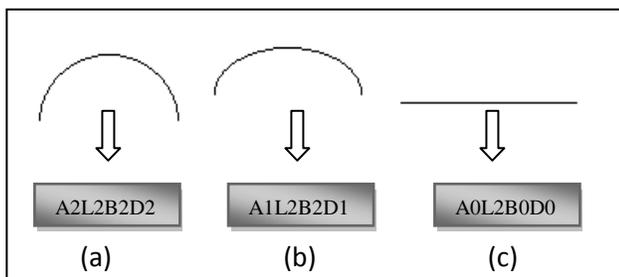
Algorithm to describe the contour in an XML file:

```
Begin

    Open an XML file F

    NP: Compute the number of points in the contour;

    NC: Compute number of curves

    Storing NC and NP in XML file F

For i=1 to NC

        C(i) ← current curve

  If (C(i) is concave) then

      Compute the values {Ai, Length li, Angle βi , Dgi }

  Else If(C(i) is convex) then

      Compute the values {Ai, Length li, Angle βi , Dgi }

  Else // Right line

      Compute the value {Length li }

  End If

    Quantifying the values of Ai, li, βi and Dgi  in its symbols.

    Storing the symbols in an XML file (F).

  End for(i)

Close the file F // Description symbolic of the shape is stored
in F

End.
```

Our algorithm creates an XML file for each shape, for example the XML descriptor computed for curve (a) illustrated in figure 4 is:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<SHAPE>
<Name>Exemple d'un descripteur XML</Name>
<C Number="1">
<TYPE>CV</TYPE>
<A>A2</A>
<l>L2</l>
<beta>B2</beta>
<Dg>D2</Dg>
</C>
</SHAPE>
```

C.   Matching shape

1)   Global matching

At this level, we are interested only in global information which characterizes the general aspect of an object. At first, matching is done with comparing the number of different components of the outline shape descriptor.

From the XML descriptor it is easy to extract the following indices:

- Number of inflection points in the contour.

- Number of curves: Computed as the number of tag <C>.

- The order of each Curve defined by its number.

- Number of convex curves: computed as the number of type CV.

- Number of concave curves: computed as the number type CC.

- Number of right lines: computed as the number of type R.

In some cases both of the contours have the same global descriptor; in this case the global matching is not possible. In this moment we use the XML descriptor for obtaining a list of candidate couples of contours that constitute the input for the next step. Using an XML descriptor reduces the execution time and the research with a large database

2)   Local matching

After the outlines are stored in XML files, their similarity can be evaluated by an appropriate comparison of each string of symbols stored in the tags <C>.

All string of symbols stored in the tags <C> are extracted from the XML file of the first shape and then compared to other strings of the other XML file of the second shape.

We use the technique of dynamic programming for a good matching between strings of symbols. The dynamic programming can find the best alignment between two strings with different lengths. When sequences of strings are aligned, sequence alignment scores are computed. The system can find similar sequences by sorting the alignment score. In this paper, we use the algorithm of Levenshtein Edit distance [14], this technique was modified by adding cost of similarity between the symbols. The edit distance between two strings is given by the minimum number of operations needed to transform one string into the other, where an operation is either an insertion, deletion, or a substitution of a single character.

We construct a matrix D[0,--,m;0,--,n]. The matrix D is computed using the recurrent equation:

$$D(i; j) = \min \begin{cases} D(i\text{-}1; j\text{-}1) + F; \ //a\ substitution \\ D(i\text{-}1; j) + w; \ //a\ deletion \\ D(i; j\text{-}1) + w; \ //an\ insertion \end{cases} \quad 2$$

D(i,j) represents the score for the matrix position, W represents a gap of penalty score its value equal to "2" and F represents the match/mismatch score.

This is a dynamic programming algorithm in Matlab language:

```
for i = 1:n1
   D(i+1,1) = D(i,1) + DelCost;
end;

for j = 1:n2
   D(1,j+1) = D(1,j) + InsCost;
end;

for i = 1:n1
   for j = 1:n2
     if s1(i) == s2(j)
       Subst = 0;
     else
       Subst = SubstCost;
     end;
D(i+1,j+1) = min([D(i,j)+Subst, D(i+1,j)+DelCost,
D(i,j+1)+InsCost]);
end;
end;
```

The first step for DP algorithm is to create a matrix with M+ 1 columns and N+ 1 rows where M and N correspond to the size of the sequences to be compared. This DP algorithm has been modified to take into account the differences resulting from the quantification of areas, distances and angles. A smaller weight or penalty (with a value lower than one) for the substitution of two adjacent symbols was introduced; for example the distance between A1 and A2 was taken to be equal to 0.5 and A1 and A3 equal to 1, and similarly the distance between B1 and B2 or L1 and L2 or D1 and D2 was taken to be equal to 0.5. The compute starting in the upper left hand corner in the matrix and finding the minimal score for each position in the matrix. The minimal score is calculated using the formula (2).

Therefore, the algorithm helps the system avoid computing an exponentially large number of comparisons. When sequences of strings are aligned, sequence alignment scores are computed.

String sequences are matched well for lower alignment scores. The system can find sequences that are similar to an query key sequence and the minimum score is selected.

Consider the sequence of the shape (a) presented in figure 4 being a query key sequence and comparing it with sequences of the two shapes (b) and (c) respectively:

The score matrix for two cases is as follows:

Contour (a) with (b)

	A2	L2	B2	D2
A1	0.5	2	2	2
L2	2	0.5	2.5	4
B2	2	2.5	0.5	2.5
D1	2	4	2.5	1

Contour (a) with (c)

	A0	L2	B0	D0
A1	0.5	2	2	2
L2	2	0.5	2.5	4
B2	2	2.5	1.5	3.5
D1	2	4	3.5	2

Fig.5: An example showing how to compute the edit distance between two strings. The last cell shows the distance computed for these two strings.

After filling the score matrix, the minimum alignment score for the sequence (a) with sequence (b) is 1 and the minimum alignment score for the sequence (a) with sequence (c) is 2, so the shape (a) is matches more with shape (b) .

III. EXPERIMENTS

The method has been tested on a set of MPEG-7 and KIMIA-25 shapes illustrated by Fig.6. For each shape, contours of objects have been extracted. After that, we determine for each contour, the inflection points using corner detection. In Fig.7 we illustrate an example of dividing the contour of the shape into a set of primitives (convex or concave curves or lines)

(a)

(b)

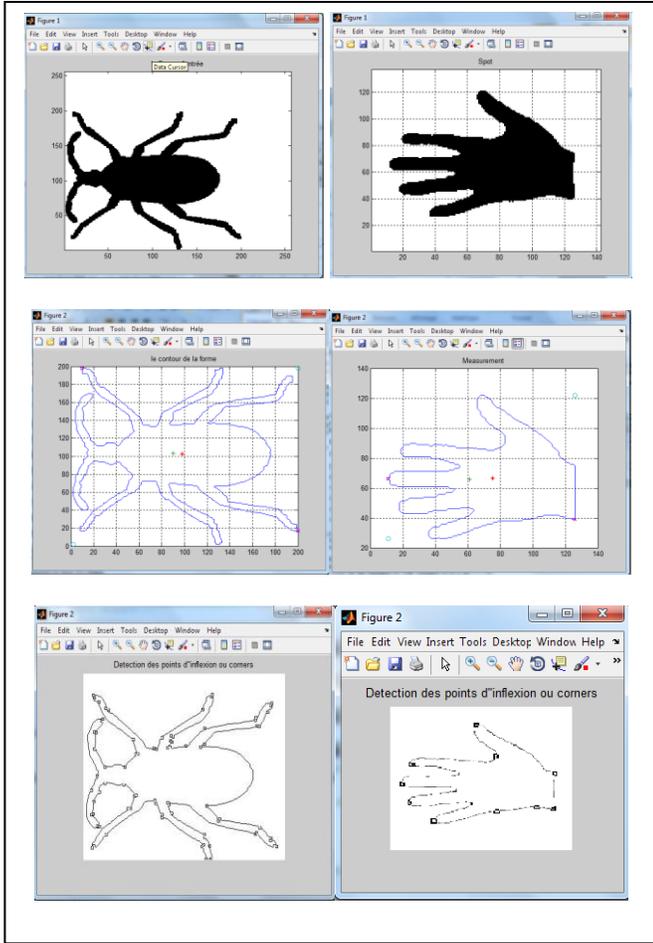Fig.6: Some of the objects in the (a) MPEG-7 database and (b) KIMIA-25 database.

Fig.7: Extraction the contour from two shapes in databases MPEG-7 and KIMIA-25 and detection of the inflection points.

The system creates an XML file for each shape and each shape is indexed by its xml file.

We took the XML file for some shapes and were used them as reference XML files for experimentation; the number of reference XML files is defined as K. The percentage of matches between the reference XML and other files is obtained, and we computed the percentage of matching for different databases and different values of K.

The number of iteration used in this paper is 10 and the different values of the parameter K can be toke range in (3, 7, 15, 20, 50).

Results are presented as a percentage. The graph in figure 8 shows the score of matching for different values of K in two databases MPEG-7 and KIMIA-25.
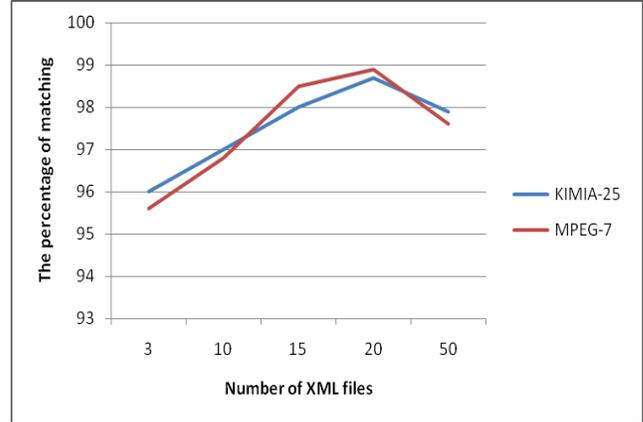

Fig.8: Percentage of matching for five values of K in MPEG-7 and KIMIA-25.

The best match is achieved for k = 20 (98.7% for KIMIA-25) and (98.9% for MPEG-7).

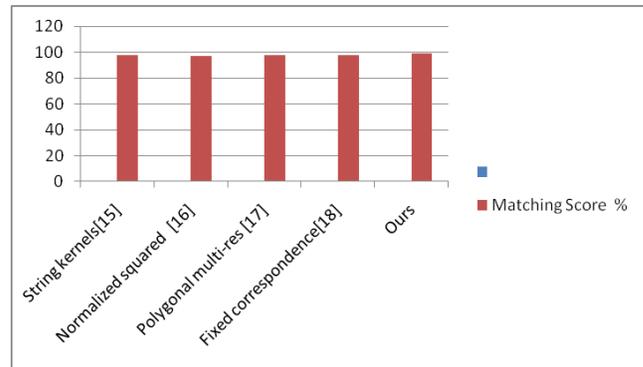These results are compared with some old methods and techniques in MPEG-7.

Fig.9: Comparison of results in MPEG-7database.

Our algorithm was compared with some old methods and our result is advanced with a little percentage compared to other solutions (Fig.9).

IV. CONCLUSION

We have presented a new technique for shape matching. A key characteristic of our approach is the transformation of the shape features into XML file and then compare these XML files using dynamic programming. After different kinds of experimentation on MPEG -7 and KIMIA-25 Shape database, the proposed method has given interesting results over the existing methods.

REFERENCES

[1] H. J. Wolfson, "On Curve Matching," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 12, pp. 483-489,1990.

[2] E. Kishon, T. Hastie, and H. J. Wolfson, "3-D Curve Matching using Splines," J. of Robotic Systems, Vol. 8, pp. 723-743, 1991.

[3] G. Barequet, and M. Sharir, "Partial Surface Matching by Using Directed Footprints," Symposium on Computational Geometry, pages C-9-C-10, 1996.

[4] I. Biederman, G. Ju, Surface versus edge-based determinants of visual recognitions, Cognit. Psychol. 20 (1988) 38–64.

[5] D. Geiger, A. Gupta, L.A. Costa, and J. Vlontzos. Dynamic Programming for Detecting, Tracking amd Matching Deformable contours. IEEE Trans. on Pattern Analysis and Machine Intelligence, 17(3):294–302, 1995.

[6] L. Floreby. A Multiscale Algorithm for Closed Contour Matching in Image Sequence. In IEEE Intern. Conf. on Pattern Recognition, pages 884–888, 1996.

[7] E.G.M. Petrakis, A. Diplaros, E. Milios, Matching and retrieval of distorted and occluded shapes using dynamic programming, IEEE Trans. Pattern Anal. Mach. Intell. 24 (11) 1501–1516. (2002).

[8] M. H. Yu, C. C. Lim, J. S. Jin, Shape similarity using XML and portal technology, Visual Information Processing VIP , Sydney, Australia 2006

[9] G. S. K. Fung, N. H. C. Yung, and G. K. H. Pang, "Vehicle shape approximation from motion for visual traffic surveillance," in Proc. IEEE 4th Int. Conf. on Intelligent Transportation Systems, pp. 201–206 _2001.

[10] G. S. Manku, P. Jain, A. Aggarwal, A. Kumar, and L. Banerjee, "Object tracking using affine structure for point correspondence,"

in Proc. 1997 IEEE Comput. Soc. Conf. on Computer Vision and Pattern Recognition, pp. 704–709 1997.

[11] B. Serra and M. Berthod, "3-D model localization using highresolution reconstruction of monocular image sequences," IEEE Trans. Image Process. 6_1_, 175–188 1997.

[12] Xiao Chen He and Nelson H. C. Yung "Corner detector based on global and local curvature properties" Optical Engineering, v. 47 n. 5, p. 057008-1 - 057008-12 (2008)

[13] N. Gherabi and M. Bahaj A new shape descriptor using XML" IJCSE Vol. 3 (1369-1376) 2011

[14] E.S. Ristad, P.N. Yianilos, "Learning string edit distance", IEEE Trans. Pattern Anal. Mach. Intell. 20 (5) 522–532 (1998).

[15] M.R. Daliri, E. Delponte, A. Verri, V. Torre, "Shape categorization using string kernels", SSPR/SPR, in: Lecture Notes in Computer Science (LNCS), vol. 4109, 2006, pp. 297–305.

[16] B.J. Super, "Learning chance probability functions for shape retrieval or classification", in: Proceedings of the IEEE Workshop on Learning in Computer Vision and Pattern Recognition, June 2004.

[17] E. Attalla, P. Siy, "Robust shape similarity retrieval based on contour segmentation polygonal multiresolution and elastic matching", Pattern Recognition 38 (12) (2005).

[18] B.J. Super, "Retrieval from shape databases using chance probability functions and fixed correspondence", Int. J. Pattern Recognition Artif. Intell. 20 (8) (2006) 1117–1137