Improving Shape Context Using Geodesic Information And Reflection Invariance

Vittal Premachandran and Ramakrishna Kakarala
School of Computer Engineering, Nanyang Technological University, Singapore

ABSTRACT

In this paper, we identify some of the existing problems in shape context matching. We first identify the need for reflection invariance in shape context matching algorithms and propose a method to achieve the same. With the use of these reflection invariance techniques, we bring all the objects, in a database, to their canonical form, which halves the time required to match two shapes using their contexts. We then show how we can build better shape descriptors by the use of geodesic information from the shapes and hence improve upon the well-known Inner Distance Shape Context (IDSC). The IDSC is used by many pre- and post-processing algorithms as the baseline shape-matching algorithm. Our improvements to IDSC will remain compatible for use with those algorithms. Finally, we introduce new comparison metrics that can be used for the comparison of two or more algorithms. We have tested our proposals on the MPEG-7 database and show that our methods significantly outperform the IDSC.

Keywords: object recognition, shape matching, shape descriptor, computer vision

1. INTRODUCTION

Shape matching is a key problem that is highly investigated in computer vision. It is well recognised that, in many cases, just the shape information is sufficient in identifying the object. In fact, objects can even be recognised when the only information that is available are the contour fragments from the object's boundary. Appearance information such as colour and texture are not important features in the recognition of such objects. Certain psychophysical studies[1] show that humans can recognize the object's class using just the object's contour. In addition, we humans are able to recognize the objects that have a characteristic shape, even if they have different colours and/or textures.

Though recognition of similar shapes is considered as a routine task for humans, it is still quite hard to achieve it in computer systems. Bronstein et al.[2] say, "All pattern recognition problems boil down to giving a quantitative interpretation of similarity (or equivalently, dissimilarity) between objects." Hence, in order to perform object recognition from just the shape information, there needs to be a way to find similarity (or dissimilarity) between shapes. A measure of similarity is usually obtained between two objects, a template and a test shape. To obtain a similarity measure, one has to first extract meaningful shape information from the template, and then match this shape information with that of the test shape. The extraction of such shape information is done by the means of shape descriptors.

The object's shape information is stored in the object's contour, i.e. the object's boundary. Hence, it makes sense to extract the boundaries of the object before performing any kind of shape matching. The boundary is usually extracted by giving the image as an input to an edge detector, which would then output the edge-map of the input grayscale image. The edge-map contains only the information from the object's shape and discards other information such as the colour and texture. After the extraction of the object's boundary, different kinds of shape descriptors are used to encode the shape information. The shape descriptors could be either local shape descriptors, or global shape descriptors. Global shape descriptors are useful when encoding rigid objects, or objects with slight articulation. On the other hand, local shape descriptors are found to be useful while obtaining partial shape similarity.

In this paper, we will look at some of the problems that traditional shape descriptors face, and propose ways to overcome them. We also show how different metrics can help improve the existing shape descriptors. The rest of the paper is organised as follows. In Section 2, we give a brief review of some of the existing work that has been done in the field of shape matching. In Section 3, we discuss one of the problems faced by the traditional shape descriptors and propose a way to solve it. Later, in Section 4, we show how changing the metrics of the shape descriptor can lead to better results. We follow this up by some experimental results in Section 5, and conclude the paper in Section 6.

Further author information: (Send correspondence to Vittal Premachandran)
Vittal Premachandran: E-mail: vittalp@pmail.ntu.edu.sg
Ramakrishna Kakarala: E-mail: ramakrishna@ntu.edu.sg

2. LITERATURE REVIEW

Most of the algorithms that perform shape matching consider edge maps as the starting source to build shape descriptors. The outer boundary of a well-segmented object can be easily obtained as an output of some edge detector. These edges are stored as a list of oriented connected co-ordinate points. With the boundaries of the shapes extracted and stored as contours, shape matching reduces to contour matching.

Different distance metrics such as the Hausdorff distance[3] and the Chamfer distance[4,5] have been used for contour matching. However, they have been found to be sensitive to clutter. Some others[6] have tried to match contours by splitting them, at points of high inflexion, into sub-fragments and then match these sub-contours. Such algorithms rely on the robustness of the corner detectors.[7] The fact that corner point detectors are not quite robust in consistently detecting the same corner points called for new means of representation of the object's contour.

Shape context, proposed by Belongie et al.,[8] eliminated the need for corner detection. The shape context requires a set of uniformly sampled points, sampled from the object's contour, without setting any restriction on the location of the sampled point. The shape context, which is a global shape descriptor that is built at each sampled point, is a 2-D histogram of distances and angles. The distances were obtained by calculating the Euclidean distance betweent the point under consideration and every other sampled point. Similarly, the angular information between any two points was obtained by finding the angle between the tangent drawn at point under consideration and the displacement vector between the two points. Ling and Jacobs[9] later modified the shape context and called it the Inner Distance Shape Context (IDSC). The distance that is used in IDSC is the "inner distance", as compared to the Euclidean distance that was used by the original shape context. The IDSC also changed the angle measurements to what they call the "inner angle". The way in which these metrics are calculated are explained in more detail in Section 4. With this modification they were able to show better performance than the original shape context.

Other shape descriptors such as the shape-tree[10] do perform well, but have high computational complexity. Due to the good performance of IDSC, and its moderate complexity, many researchers have used it in their work to extend the field of shape matching. Kontschieder *et al.*[11] perform post processing on the results obtained from IDSC to obtain better recognition rates, while Gopalan *et al.*[12] process the data before hand, and then use IDSC for recognition. In either of these cases, IDSC remains as the core matching procedure.

As good as IDSC is considered to be at this point, we feel that it can be further improved from the point of view of computational complexity and also in the way it chooses its shape descriptors. In the following section, we discuss how we can bring down the computational complexity and in Section 4, we show how a different distance and angle metric can help in boosting the recognition rates.

3. INVARIANCE

Both the IDSC, and the original shape context, sample the contour, uniformly, into N discrete points. These points are arranged in a clockwise fashion to have a consistency in representation, and to preserve the adjacency information. However, the restriction to represent the contour in a clockwise direction brings about a problem when the a shape is to be matched to its reflection. We consider the reflection across the X-axis and Y-axis as one and the same, because the reflection across Y-axis (or X-axis) can be considered as the reflection across the X-axis (or Y-axis) followed by a rotation, and both the shape context and the IDSC are invariant to rotation.

Consider the example of a shape A, which is denoted by the set of sampled points, $A_s = \{p_1, p_2, ..., p_N\}$, arranged in a clockwise fashion. If the shape were to be reflected to, say $A^{'}$, and then given as an input to the sampling algorithm, we would then obtain the set of sampled points as $A_s^{'} = \{p_N^{'}, p_{N-1}^{'}, ..., p_1^{'}\}$ (or a cyclic permutation of $A_s^{'}$), where $p_i^{'}$ is the reflected version of point p_i. The order of the points is reversed, as compared to A_s, so that the set of points still represent the shape in a clockwise direction.

When the shape A is now matched to shape $A^{'}$, point p_i gets matched with point p_{N-i+1}, which causes the matching algorithm to output a high matching cost. Such high costs fool the algorithm into thinking that the test and template objects are not similar in shape. Since the shape context is not reflection invariant, in order to get the true cost in the cases of reflection, the matching has to be repeated on a reflected version of the test image as well. Thus, we get the true cost of matching two shapes, say A and B, as given below

$$\Psi(A, B) = min(\Gamma(A, B), \Gamma(A, B')). \tag{1}$$

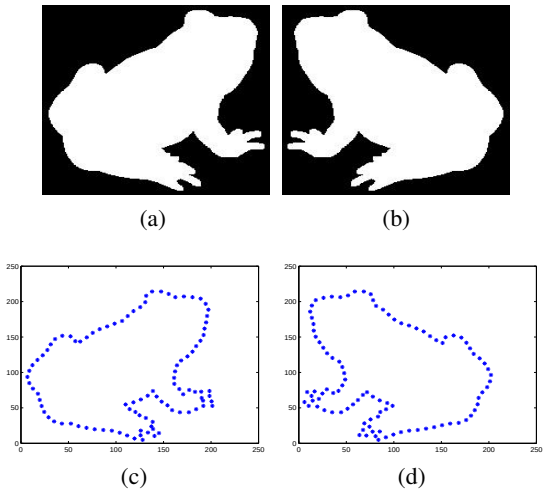


Figure 1: (a) and (b) are reflections of each other. (c) and (d) are the plots of the sampled points from the contours of (a) and (b) respectively.

Here, $\Gamma(A, B)$ is the cost obtained from a shape matching algorithm, $B^{'}$ is the reflected version of the shape B, and $\Psi(A, B)$ is the "true distance" between the two shapes A and B. From Equation 1, it is clear that in order to obtain the "true cost" of matching two shapes, the shape-matching algorithm has to be run twice. Ideally, we would not like to perform this matching twice, as it doubles the time required to get the cost between two shapes. Hence, it would be desirable if we could prevent this duplicate matching by transforming all the shapes into some canonical form before obtaining the shape descriptors and performing the matching. To our knowledge, there has been no work done to achieve such a reflection invariance with Shape Context. We now go on to propose a method, which helps in transforming the images to a canonical form. The efficiency of this algorithm will be discussed in Section 5.

Figures 1a and 1b show two images, which are reflections of each other. The reflection is quite apparent to us because of the orientation of some of the "characteristic" parts of the object. However, it is difficult to articulate what this "characteristic" property actually is. Hence we use the statistics obtained from the image contours to make the decision. The major property that we use, to perform reflection normalisation, is that the object's contour is uniformly sampled and that same number of sampled points are extracted from all the images. We first take the sampled points from each image and normalise their centroids. This helps us to make the decisions on the orientation of the objects. We can now say that an object that is symmetric across both the X and Y axis has the same number of sampled points in each of the four quadrants. Secondly, for an object that is symmetric across the X-axis (or the Y-axis), the number of points on either sides of the X-axis (or the Y-axis) remains the same. Finally, for an asymmetric object, the number of sampled points is not going to be the same across either of the two axes.

An object can be brought to its canonical form if it were to be aligned in such a way that one of the quadrants always has the maximum number of sampled points. Without loss of generality, we choose the first quadrant for this purpose. Before we can proceed with normalising the object's orientation based on reflection, we have to normalise it based on rotation. Objects that have been rotated to different degrees can have varied number of sampled points in each of the quadrants. To perform rotation normalisation, we use the Principal Component Analysis (PCA).[13] The PCA, for a 2-D data set, orients the data along the two dominant axes. We use the sampled points, obtained from the contour, as the input to the PCA.

We set the eigen vector, corresponding to the dominant eigen value, as the X-axis of the normalised shape. The final step is to align the objects in such a way that the first quadrant always has the maximum number of sampled points. This alignment can be done by simple reflections of the rotation-normalised objects, across the X-axis and/or the Y-axis. The procedure is formalised below.

Let us consider P to be a mean-subtracted 2xN matrix, where the columns represent different sampled points, and the rows contain the co-ordinates of the sampled points. The dominant directions are obtained from the eigen vectors of the covariance matrix of the input data, P. The eigenvectors can be obtained using the eigenvalue decomposition of the

covariance matrix, as shown below

$$A \quad = \quad PP^T \quad = \quad Q\Lambda Q^T \tag{2}$$

The rotation-normalised shape matrix is given as

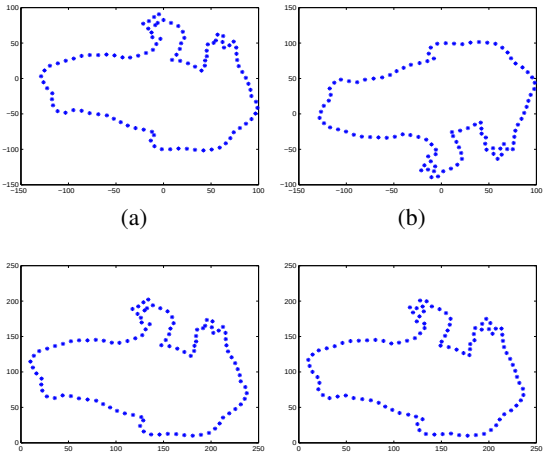
$$P_{rot} = Q^T P. \tag{3}$$

Figures 2a and 2b show the contour points after rotation normalisation. We now count the number of sampled points that fall in each of the four quadrants and perform reflection normalisation. Algorithm 1 shows how we perform reflection normalisation.

Algorithm 1 Algorithm to show how reflection normalisation can be performed.

$q := getQuadrantWithMaximumSamples(P_{rot})$;
$R_y := [-1, 0;\ 0, 1]$;
$R_x := [1, 0;\ 0, -1]$;
if q == 1
    $Q_1 := Q$;
elsif q == 2
    $Q_1 := R_y Q$;
elsif q == 3
    $Q_1 := R_x R_y Q$;
elsif q == 4
    $Q_1 := R_x Q$;
fi
$P_{ref} := Q_1^T P_{rot}$;

Figures 2c and 2d show the contour points after reflection normalisation. We can see that both the objects are now oriented in the same direction. In Figure 3 we can see the effect of reflection normalisation on the objects of a whole class.



(a) (b) (c) (d)

Figure 2: (a) and (b) show the contour points from Figures 1c and 1d after rotation normalisation. (c) and (d) show the contours after reflection normalisation.

4. GEODESIC SHAPE CONTEXT

In the previous section, we saw how reflection invariance could be achieved by bringing all the shapes to a certain canonical form, and why it was necessary to do so. In this section, we look at how the IDSC's core matching procedure could be enhanced by the use of better descriptors. The two main ingredients of the shape context were the distance information

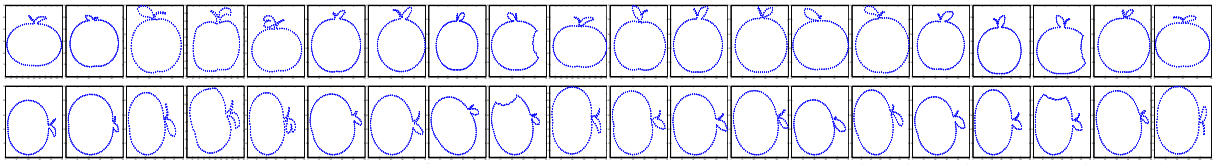


Figure 3: The top row shows all examples from a class 'apple'. We can see that their orientations are quite random. The bottom row shows the same images after reflection normalisation. It can be seen that all images (except the last) are now in a standard canonical form.

and the angular information. While the original authors of the shape context[8] suggested the use of Euclidean distance for calculating the distance between the two points, Ling and Jacobs[9] pointed out that such a distance measurement was not sufficient in capturing the complete shape information in a given shape. By using the Euclidean distance, we lose information about the shape of the contour joining the two points. Hence, Ling and Jacobs proposed a new shape context, namely, the Inner Distance Shape Context (IDSC). The IDSC is also made of the same two ingredients, the distance and the angle. However, it differs in how the distance and the angle values are calculated.

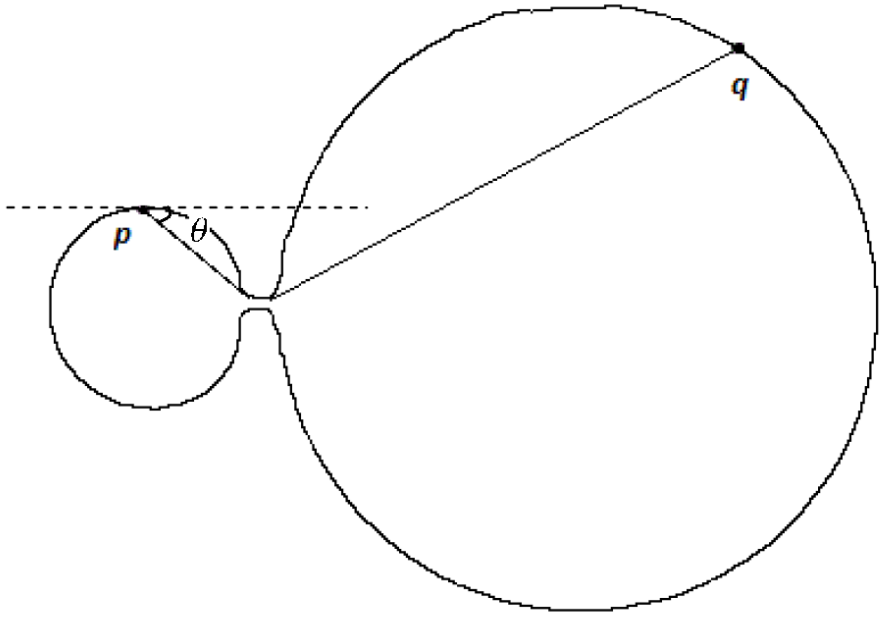


Figure 4: The inner distance between any two points, p and q, is the shortest distance along the path joining the two points, such that the path lies completely within the boundary of the object. The inner angle is the angle between the tangent at the first point and the first part of the path leading from the first point, p, to the second point, q.

For any two given points, p and q, the inner distance between the two points is the shortest distance along the path joining the two points, such that the path lies completely within the boundary of the object, and, the angle between them is the angle between the tangent drawn at point p, and the path leading to point q. The authors name this angle as the "inner angle". Figure 4 gives an example of how the inner distance and the inner angle are calculated.

While the inner distance captures much more of the object's geodesic information than the Euclidean distance, the usage of the inner angle leads to loss of vital information. Take for example the objects given in Figure 4. The figure shows an object that is clearly divided into two parts, the left lobe and the right lobe. The problem that we face while using the

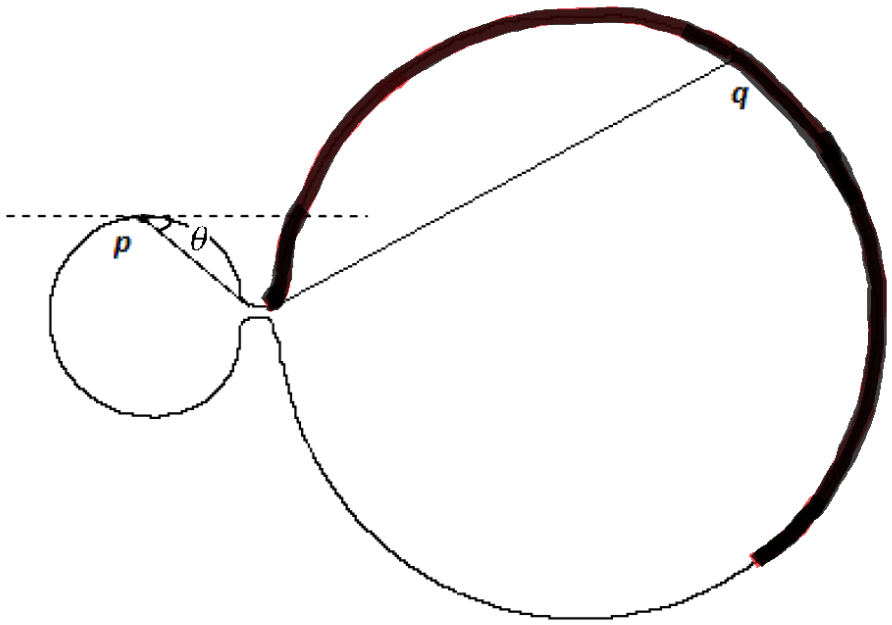


Figure 5: The inner angle between the point, p, and all other sampled points from the contour segment that has been thickened, will have the same value as that between the two points, p and q. The parts of the right lobe that are not thickened will have a different inner distance path, w.r.t the point p, and will therefore have a different inner angle.

inner angle is that we get the same angular values for the angles between a particular point, say p, on the left lobe, and a lot many of the points on the right lobe. This results in a loss in information regarding the exact location of those points in the right lobe, with respect to point p. Figure 5 gives an illustration of the same.

Hence, the first modification to IDSC that we propose brings about a change in the way the angle is measured. The angle between two points, say p and q, in our implementation, is calculated as the angle between the tangent at point p, and the displacement vector between the two points, starting from point p and ending at point q. We name this angle as the Euclidean Angle. Since a tangent at a point is a line extending in both directions, there might be an ambiguity in the way the angle is actually calculated. We eliminate the ambiguity by assigning directions to the tangents. The sampled point at which the tangent is drawn is considered as the local origin and the tangent at that point is considered as the X-axis. The part of the tangent, in the direction of the next sampled point, is considered as the positive X-axis. This eliminates the ambiguities that arise while calculating the angles.

Figure 6, which is a modification of Figure 4, gives an illustration of how the "Euclidean Angle" is calculated. The tangent at point p, in Figure 4, is shown by the use of a dotted line. The tangent extends in both the directions and hence there arises an ambiguity while calculating the "inner angle". In Figure 6 we show show how this ambiguity can be eliminated by giving a direction to the tangent at point p. The black arrow shows the positive direction of the X-axis. The "Euclidean Angle" between p and q is the angle that the displacement vector (the red arrow) makes w.r.t. to the positive direction of the local X-axis.

Secondly, we also modify the way in which the distance metric is calculated. We propose to replace the use of both the Euclidean distance[8] and the inner distance,[9] with the geodesic distance. The use of the Euclidean distance in the original shape context gave the overall displacement between any two points. More of the shape information was embedded, in the IDSC, by the use of inner distance. However, since the inner distance between any two points is the length of the shortest path between them, such that the path lies completely within the shape, it cannot fully capture the manifold properties. Hence, we propose to use the geodesic distance, which does capture these manifold details.

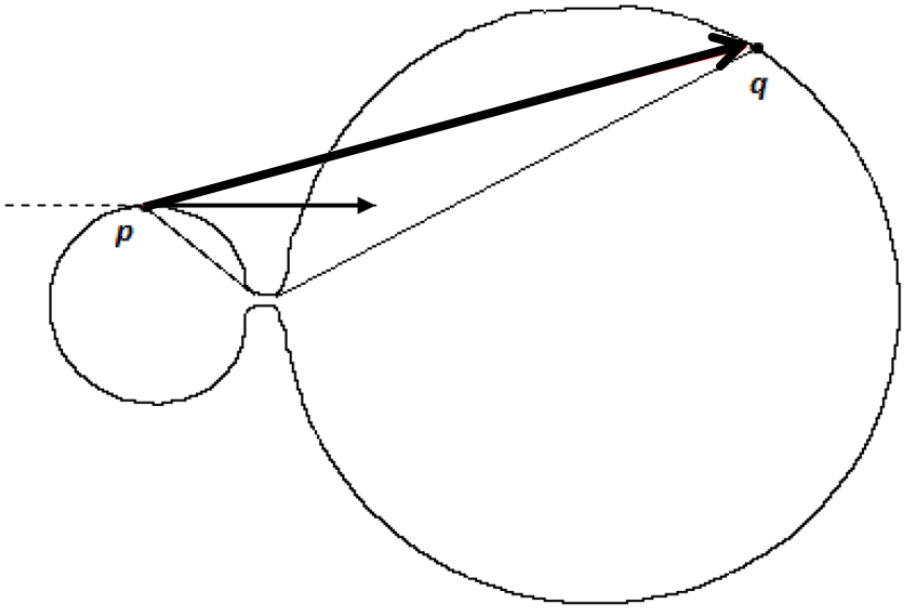


Figure 6: The Euclidean angle is calculated as the angle between the tangent (shown by the thin arrow) at a point p and the displacement vector (shown by the thick arrow), starting at point p and ending at point q.

The geodesic distance between any two points is the shortest distance along the contour joining the two points. For any two points on the contour, p and q, the geodesic distance is approximated to be the sum of the linear distances between consecutive sampled points, starting from point p and ending at point q. The geodesic distance, just like the inner distance, is invariant to articulations. Though, theoretically, the Euclidean Angle is not invariant to articulations, we can see, from the experimental results in Section 5, that the Euclidean angle can withstand "mild articulations". The reason for this, we believe, is because of the bin size that is used while obtaining the histograms. We use 12 angular bins (same as IDSC), which allow the angles to have a variance of ± 15 degrees.

In the next section, we show the impact of these changes that is evident from our experiments.

5. EXPERIMENTS AND RESULTS

In order to test our descriptor, we perform our experiments on the well-known MPEG-7 dataset.[14] The MPEG-7 database consists of 1400 images in total. The database consists of images from 70 different classes. Each class consists of 20 images. The MPEG-7 dataset is a challenging dataset as the objects in each class not only differ by translation, scale, and rotation, but are also deformed and occluded in many cases. Figure 7 shows an example from each of the 70 classes, in the database. Before we look at the performance of the reflection invariant, we first define some of the scores that are used for comparison and see how the modifications from Section 4 affect the same. Then, in Section 5.1, we analyse the performance of our reflection invariant.

The recognition rate for the MPEG-7 database is measured by a score known as the Bullseye score. To compute the bullseye score, each image is matched with every other image in the database, which are then sorted in ascending order, based on the cost of matching. The top 40 images, i.e. 40 images with the least cost of matching, are obtained. At most, 20 of these 40 images can belong the same class as the query image. If all 20 images from the same class as the query image are present in the top 40 best matches, then the bullseye score for that query image is said to be 100%. However, if the top 40 best matches did not contain all 20 images from the same class, the bullseye score is $(k/20) * 100$, where k is

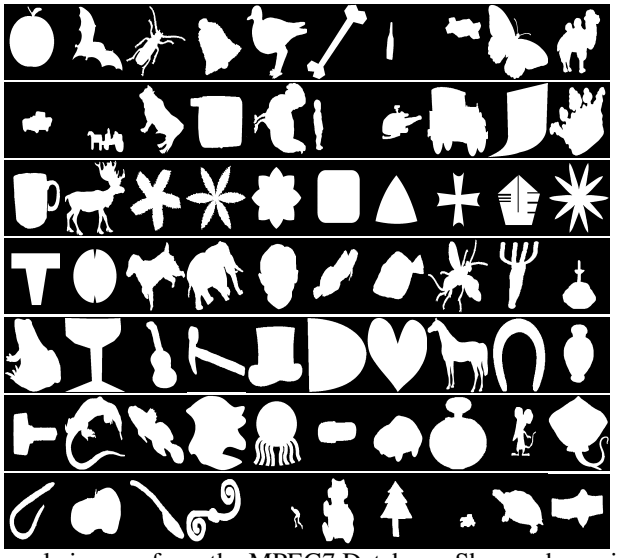


Figure 7: The figure shows example images from the MPEG7 Database. Shown above, is an example from each class.

the number of objects from the same class as the query image, in the top 40 best matches. Table 1 shows a comparison of the Bullseye scores obtained from other algorithms. Note that others have published higher bullseye scores, but that is only after pre-processing the database and then using IDSC,[12] and/or post-processing on the matrix obtained from IDSC.[11] Our algorithm will still remain compatible with these pre/post processing algorithms.

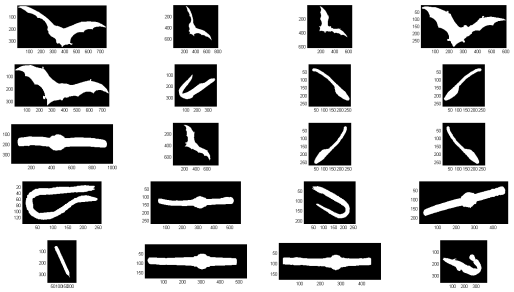
Algorithm	Bullseye Score
SC+TPS[8]	76.51%
Curve Edit[15]	78.17%
IDSC[9]	85.40%
Ours	**86.73%**

Table 1: The table shows the Bullseye scores obtained from other algorithms in comparison with the Bullseye score obtained from our algorithm.

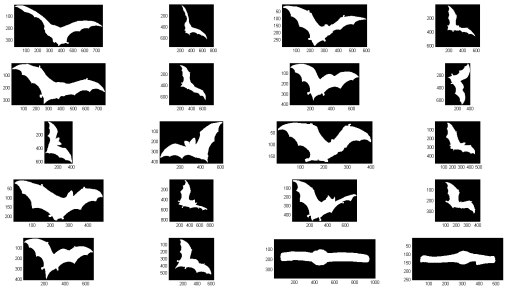
We set the following parameters in our experiments, which is the same as that set in Ling & Jacobs.[9] The number of sample points, $N = 100$, the number of distance bins, $N_d = 8$, the number of angular bins, $N_\theta = 12$, and the number of locations to search for the points alignment is, $k = 8$. We also account for the handling of the mirror image examples by comparing each image to every other image as well as the reflected version of every image in the database. Figure 8 shows the comparison of retrieval results of our method alongside the method of Ling and Jacobs.[9]

We have also tested the impact of the modifications, independently. Of the two changes that were proposed in Section 4, the change from "inner angle" to the "Euclidean Angle" produce the maximum increase in performance. With the introduction of just the "Euclidean Angle" (and retaining the distance metric as was proposed in IDSC) the bullseye score increases from 85.40%[9] to 86.49%. With the introduction of the geodesic distance as a distance metric, the Bullseye score increases even further to 86.73%.

The performances of the modified algorithm and IDSC, over the whole database, can be visualised in Figure 9. The graph shows the bullseye scores for individual classes. The X-axis represents the classes and the Y-axis represents the bullseye score. The maximum score that each class can have is 1.0. If a particular class has a bullseye score of 1.0, it means that each image of that class, when used as the query image, was able to retrieve every other image from its class among the 40 best matches. The blue line shows the performance of the original IDSC and the red line shows the performance of our algorithm. It can be seen that the red line is above the blue line, in a majority of the cases. The modifications introduced enable us to obtain better, or at least equal Bullseye scores, as that of the original IDSC, among 50 of the 70 classes.



(a)



(b)

Figure 8: (a) The figure shows the retrieval results of the original IDSC algorithm[9] when a bat was given as an input. Of the 20 best matches, only 6 of them belong to the same class as the query image. (b) The figure shows the retrieval results of our algorithm when the same bat was given as an input. Of the 20 best matches, 18 of them belong to the same object class as the query image.

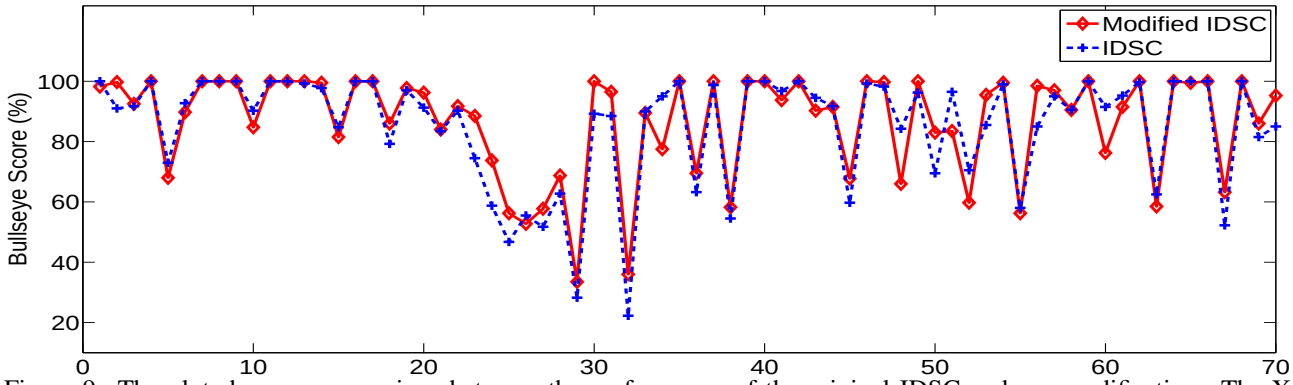


Figure 9: The plot shows a comparison between the performance of the original IDSC and our modification. The X-axis represents the different classes and the Y-axis represents the Bullseye Score. From the plot, we can see that our modifications perform better than, or at least as good as, IDSC in 50 of the 70 classes.

The bullseye score, which was explained previously in this section, is quite lenient towards the algorithms in two different ways. The first leniency is apparent from the range that is allowed for checking the presence of objects from the same class. The bullseye score is checked over the 40 best matching images. We say that the range, i.e. 40, is fairly lenient because there are only 20 images in each class. So, ideally, we would like to compute the Bullseye score over this restricted set of 20 best matches. Hence, we introduce a new metric, named the 'Ideal Bullseye Score', which is calculated over a restricted (or ideal) range of 20 best-matching images. When the 'Ideal Bullseye Score' was used to calculate the efficiency of retrievals, we found that the original IDSC algorithm had an 'Ideal Bullseye Score' of 77.07% and our modifications to the IDSC boosted the 'Ideal Bullseye Score', by 2.25%, to 79.32%. This shows that not only are more correct images being retireved using our shape descriptors, more of these correct images are located among the best 20 retrieved images.

The second leniency that we claim that the bullseye score provides is that it assumes equal ranks to locations of the

retrieved images. It does not credit, or penalise, the algorithm based on the locations of the retrieved images. If an algorithm, say A, retrieves all 20 images belonging to the same class as the query object, in the first 20 locations of the top 40 best matches, its bullseye score will be same as another algorithm, say B, which would also retrieve all 20 images belonging to the same class as the query object, but in the final 20 locations among the best 40 retrieved images. Clearly, this is a problem with the metric that is used to test the efficiency. Hence, we introduce another new metric for comparison of the algorithm's efficiency and name it as the 'MPEG-7 Rank'. When a particular image is used as a query image for object retrieval, the algorithm performs shape matching and gives out the costs of matching the query shape with every other shape in the database. In order to calculate the MPEG-7 Rank, we first arrange these costs in an ascending order and obtain the locations of the 20 images belonging to the same class as the query object. These locations, or indices, denote how "far" the objects are from the query shape. The sum of all these indices would give the 'MPEG-7 Rank' of the query object.

Since there are 20 objects in each class, the best locations that the objects can take would be locations 1 to 20. This would mean that the best MPEG-7 Rank for any query object would be $\sum_{N=1}^{N=20} N$, which is equal to 210. In order to make the best possible MPEG-7 Rank to be the Rank 1, we normalise the MPEG-7 Rank by subtracting a value of 209 from the sum of the indices. Lower the MPEG-7 Rank, better is the algorithm's performance.

Henceforth, whenever we use the term 'MPEG-7 Rank', we are referring to the rank after normalisation. The MPEG-7 Rank for the whole database can be calculated as the average of the MPEG-7 Ranks of the individual objects. So, if an algorithm's Ideal Bullseye Score is 100%, its MPEG-7 Rank would be 1.

As a final metric for comparison, we use the MPEG-7 Rank to compare the performance of the IDSC's shape descriptors versus our shape descriptors. The IDSC algorithm gives an average MPEG-7 Rank of 943. On the other hand, the algorithm with our shape descriptors gives an average MPEG-7 Rank of 651. This, clearly, shows that the shapes retrieved using the modified descriptors are "closer" to the query image (cost-wise), and hence can be said to have better retrieval efficiency than the IDSC. We summarise our results in Table 2.

Algorithm	Bullseye Score	Ideal Bullseye Score	MPEG-7 Rank
IDSC[9]	85.40%	77.07%	943
Ours	**86.73%**	**79.32%**	**651**

Table 2: The table shows a comparison between IDSC and our algorithm over the three different metrics. The scores that are listed in this table are calculated from the cost matrix, which is populated with the "true-cost" (Equation 1).

5.1 Invariance

Now that we have introduced these new metrics for comparison, we use the same to test the performance of our reflection invariant. The MPEG-7 database, has classes whose objects are symmetric as well as classes whose objects are asymmetric. The database, as is, is slightly biased towards one of the orientations; i.e. in a class, the number of objects "facing" a particular direction are more than the number of objects "facing" the other direction. Hence, in order to make the database unbiased, we randomly flipped the images. The database was now spread evenly, across the two orientations, among the asymmetric classes.

This randomly-flipped image database was now used to obtain the Bullseye Score, the Ideal Bullseye Score, and the MPEG-7 Rank. First of all, we make it clear that the flipping of the images, randomly, had no effect on any of the scores when each image was compared with every other image and their reflections, which is expected. However, when the images were compared only once, we were able to see a significant drop in the scores. We then transformed every image in the database to a canonical form, using the invariance technique described in Section 3 and repeated the experiments using single matching. When the canonical database was used, we could see a significant improvement in the scores as compared to the scores that were obtained when we used the randomly flipped database. Table 3 tabulates the different scores to show the improvements in the same, which were caused by the use of our invariance. We emphasise, again, that the scores in Table 3 should not be confused with the scores in Table 2. The ones in the Table 3 are obtained using the single comparison, while the ones in Table 2 were obtained after double comparison, as given in Equation 1.

Algorithm	Bullseye Score	Ideal Bullseye Score	MPEG-7 Rank
IDSC Without Invariance[9]	74.20%	64.58%	1430
Ours Without Invariance	**75.10%**	**65.75%**	**1353**
Ours With Invariance	**78.86%**	**71.30%**	**1151**

Table 3: The table shows the improvements in the scores when the invariance is used while performing **single** comparison. It should not be confused with the scores in Table 2, which are obtained after double comparisons.

6. CONCLUSION

We make three contributions in this paper. Firstly, we point out the need for reflection invariance and propose a method to achieve the same. The invariance technique brings all the images in a database to a canonical form, which can then be used for shape matching. Secondly, we point out some of the drawbacks of the shape descriptors of IDSC, a well-known algorithm, which has been used by many researchers as the base algorithm for shape matching. We propose new shape descriptors and show how they can overcome the identified drawbacks. Thirdly, we identify some of the flaws in the metrics that were used to compare the algorithm's efficiency and propose new metrics to overcome the same. We also show, experimentally, how the proposals in this paper outperform IDSC. Our future work would be directed towards the development of better reflection-invariance techniques.

REFERENCES

[1] Biederman, I. and Ju, G., "Surface versus edge-based determinants of visual recognition," *Cognitive Psychology* **20**(1), 38–64 (1988).

[2] Bronstein, A., Bronstein, M., Bruckstein, A., and Kimmel, R., "Partial similarity of objects, or how to compare a centaur to a horse," *International Journal of Computer Vision* **84**(2), 163–183 (2009).

[3] Huttenlocher, D., Klanderman, G., and Rucklidge, W., "Comparing images using the hausdorff distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence* , 850–863 (1993).

[4] Brogefors, G., "Hierarchical chamfer matching: A parametric edge matching algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence* , 849–865 (1988).

[5] Shotton, J., Blake, A., and Cipolla, R., "Multiscale categorical object recognition using contour fragments," *IEEE Transactions on Pattern Analysis and Machine Intelligence* , 1270–1281 (2007).

[6] Petrakis, E., Diplaros, A., and Milios, E., "Matching and retrieval of distorted and occluded shapes using dynamic programming," *IEEE Transactions on Pattern Analysis and Machine Intelligence* , 1501–1516 (2002).

[7] Tissainayagam, P. and Suter, D., "Assessing the performance of corner detectors for point feature tracking applications," *Image and Vision Computing* **22**(8), 663–679 (2004).

[8] Belongie, S., Malik, J., and Puzicha, J., "Shape matching and object recognition using shape contexts," *IEEE Transactions on Pattern Analysis and Machine Intelligence* , 509–522 (2002).

[9] Ling, H. and Jacobs, D., "Shape classification using the inner-distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence* , 286–299 (2007).

[10] Felzenszwalb, P. and Schwartz, J., "Hierarchical matching of deformable shapes," in [*Computer Vision and Pattern Recognition*], 1–8 (2007).

[11] Kontschieder, P., Donoser, M., and Bischof, H., "Beyond pairwise shape similarity analysis," *Asian Conference on Computer Vision* , 655–666 (2010).

[12] Gopalan, R., Turaga, P., and Chellappa, R., "Articulation-invariant representation of non-planar shapes," *European Conference on Computer Vision* , 286–299 (2010).

[13] Duda, R., Hart, P., and Stork, D., [*Pattern Classification and Scene Analysis 2nd ed.*] (1995).

[14] Latecki, L., Lakamper, R., and Eckhardt, T., "Shape descriptors for non-rigid shapes with a single closed contour," *Computer Vision and Pattern Recognition* , 424–429 (2000).

[15] Sebastian, T., Klein, P., and Kimia, B., "On aligning curves," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **25**(1), 116–125 (2003).