Adaptive Locally Affine-Invariant Shape Matching

Smit Marvaniya, Raj Gupta, Anurag Mittal

Computer Science and Engineering Department,
Indian Institute of Technology Madras,
Chennai, INDIA - 600036.

Abstract

Matching deformable objects using their shapes is an important problem in computer vision since shape is perhaps the most distinguishable characteristic of an object. The problem is difficult due to many factors such as intra-class variations, local deformations, articulations, viewpoint changes and missed and extraneous contour portions due to errors in shape extraction. While small local deformations has been handled in the literature by allowing some leeway in the matching of individual contour points via methods such as Chamfer distance and Hausdorff distance, handling more severe deformations and articulations has been done by applying local geometric corrections such as similarity or affine. However, determining which portions of the shape should be used for the geometric corrections is very hard, although some methods have been tried. In this paper, we address this problem by an efficient search for the group of contour segments to be clustered together for a geometric correction using Dynamic Programming by essentially searching for the segmentations of two shapes that lead to the best matching between them. At the same time, we allow portions of the contours to remain unmatched to handle missing and extraneous contour portions. Experiments indicate that our method outperforms other algorithms, especially when the shapes to be matched are more complex.

Keywords: Shape Matching, Shape Retrieval, Contour Segmentation, Occlusion Handling.

Email addresses: smit@cse.iitm.ac.in (Smit Marvaniya), gupta.raj@ieee.org (Raj Gupta), amittal@cse.iitm.ac.in (Anurag Mittal)

URL: www.cse.iitm.ac.in/~amittal (Anurag Mittal)

1. Introduction

Matching of deformable object shapes is an interesting as well as an important problem in Computer Vision since shape is one of the most distinguishing characteristics of an object, being unaffected by photometric changes and background variations. Furthermore, it has been found from human perception that, in the presence of challenges such as partial occlusions, local articulations, geometric distortions, intra-class variations and viewpoint changes, it is possible to identify and recognize an object simply from its shape. Thus, shape matching has been successfully used in various tasks such as Object Detection and Classification ([1, 2, 3, 4]), Optical Character Recognition [5], Medical Image Registration [6] and Image Retrieval [7].

However, the task of modeling such variations as mentioned above in a computer is quite challenging. Furthermore, in real scenarios, when an object is segmented out automatically using techniques such as Background Subtraction or Image Segmentation, the matching of the extracted contours should be robust to errors introduced by the segmentation process. For instance, the output of a Background Subtraction technique often misses out some portions of the object or adds some extra portions such as object shadows. Sometimes, two objects may be merged into one if they are close to each other. Similar problems exist due to the use of Image Segmentation techniques as well. Figure 1 shows the output obtained from standard Image Segmentation algorithms of Russell et al. [8] and Brox et al. [9] respectively. Note that the legs of the horse are combined together in Figure 1(a) whereas some of the leg portions of the horse are missed out in Figure 1(b). A robust shape matching algorithm must deal with such variations and distractions in order to be useful in a practical scenario.

Several algorithms have been considered in the past for the problem of shape matching. As in [10], most of these can be broadly classified into two categories based on the types of features used: 1. methods that treat the shape as a blob in order to come up with an approximate representation, and 2. methods that use the contour boundary information directly.

One of the most popular blob-based approaches is to use a shock graph or a medial-axis transform for shape representation. Techniques that use these ([11, 12, 13, 14, 15]), first build a graph that models the skeleton of a shape. Topological similarity between the graphs helps in identifying the global shape structure, whereas geometric similarity at every node helps to

(a) (b)

Figure 1: Some segmentations obtained from the standard Image Segmentation algorithms of Russell et al. [8] and Brox et al. [9] respectively.

capture the local shape information. These methods perform well in the presence of deformations. However, they build the skeleton *a-priori* and can only match shapes when there is an overall global similarity between them and may fail in the presence of articulations, occlusions or noise in shape extraction.

To deal with such challenges some methods ([16, 17]) segment the shape into regions or parts that are then used for the task of matching. These methods capture the local shape variation much better by allowing articulations of such portions about each other. Felzenszwalb [16] proposed a technique for shape representation based on triangulated polygons and used Dynamic Programming to match such representations. However, this method can lead to errors in the presence of occlusions. To handle partial matching of shapes, Bronstein et al. [17] proposed a *pareto* framework to determine an optimal tradeoff between part similarity and part decomposition. Although the ideas in this paper are quite close to our work, the method is computationally expensive due to working on shape blobs. Furthermore, the optimization method proposed to search over the parts is computationally very expensive and gives only an approximate solution.

While it may be claimed that blob-based methods are more robust due to the consideration of the entire 2D space, such methods tend to be computationally very expensive due to the processing of all the pixels enclosed by a shape. Thus, it is much more efficient to use the boundary information alone in order to match shapes.

Methods that use only contour boundary information for shape representation can further be classified into 1. Global 2. Part-based methods.

Many global methods exist such as shape descriptors ([5, 18, 19]), shape distances ([20, 21]) and contour matching techniques ([22, 7, 23, 24, 25, 26, 27]), some of which also estimate the affine or projective transformation required to match the shapes [5, 27]. Among these, Shape Context [5] is a popular method that builds a shape descriptor using the Euclidean distance and the relative orientation of the contour points in a log-polar space. The dissimilarity between two shapes is a weighted sum of the matching errors, computed using a maximum bipartite algorithms and a measure on the transformation required to match the two shape contours. The method works well to match shapes invariant to rigid transformations and deals with small deformations present in the contour boundary. One of the popular extensions of Shape Context (proposed by Mori et al. [18]) solves the problem of shape matching very efficiently using multi-stage pruning techniques. The first stage is called representative shape contexts that matches very few shape contexts and identifies the outliers very fast, whereas the second stage matches the shapes in more details based on vector quantization in the space of shape contexts that involves clustering of the vectors, called as shapemes. All these methods rely on global features and hence fail in the presence of articulations, partial occlusions and noise present in the contour boundary.

To address such problems, Hong et al. [28] and Adamek and O'Connor [29] represent shapes in terms of local features such as concave or convex portions of a contour to preserve the local geometry. Even though these more local methods are robust to some deformations, articulations and noise, they do not preserve sufficient contour information for a very discriminative matching. To model shapes better, the techniques mentioned in [30] and [31] combine local and global features. Felzenszwalb and Schwartz [30] proposed a hierarchical matching technique for deformable shapes even in the presence of a cluttered background wherein a tree is built whose leaf nodes capture the local information and nodes close to the root capture global information. A Dynamic Programming based matching technique is used to match the two *shape trees*. On the other hand, Xu et al. [31] proposed a *Contour Flexibility* descriptor that gives a deformation potential to each contour point so as to deal with deformations. The similarity between shapes is calculated by considering a linear combination of the local and the global measures. Although these methods use both local and global features and perform well against deformations, they do not consider partial matching of shapes and so may fail in the presence of occlusions.

In order to handle occlusions, Latecki et al. [23] developed an elastic

shape matching algorithm based on an efficient Dynamic Programming based matching approach. This method identifies the outliers that may be present in the query shapes by allowing skips while matching. However, this method fails to capture the part structure of a shape and so may perform poorly in the presence of articulations.

There have been numerous research efforts ([32, 33, 34, 35, 24]) to deal with the local shape variations of an object shape and solve the problem of articulations. Cao et al. [24] proposed an approach for matching shape contours using the "procrustes" distance between shapes [36] and handles occlusions and shape segmentation by an MCMC (Markov chain Monte Carlo)-based search for the matching segments in two contours. However, this method is computationally quite expensive due to the consideration of individual point-point matchings across the shapes and the MCMC iterations required for optimizing such point-point matchings for the whole shape. Ma et al. [35] proposed a technique for partial matching using geometric relations of shape context as shape descriptor followed by maximal clique inference based hypothesis used to identify the best possible part correspondences. Although, this method handles the problem of partial occlusions, it may fail in the presence of articulations as the local descriptors are not restricted to capturing information only within a part. Furthermore, the method is computationally very expensive due to the use of sampling methods.

Another popular approach for handling articulations is the Inner Distance Shape Context (IDSC) proposed by Ling and Jacobs [37] that solves the problem of articulation in certain scenarios and can be considered as an improvement over Shape Context [5]. This method builds a descriptor based on the relative spatial distribution of the contour points using the Inner Distance instead of the Euclidean distance, and the Inner Angle instead of the regular angle. The Inner Distance (ID) between a pair of contour points is defined as the length of the shortest path between them while totally remaining within the shape and the Inner Angle is the angle from one point to the other that is in the direction of this shortest "inner" path. A Dynamic Programming-based algorithm instead of a bipartite matching approach was also introduced by taking advantage of the ordering constraint in order to solve the point correspondence problem. This method is invariant to the 2D-articulations of a shape as it captures the part structure effectively. However, it is not invariant to affine changes of individual parts and also fails under partial occlusions as all the contour points are considered while building the descriptor and while matching.

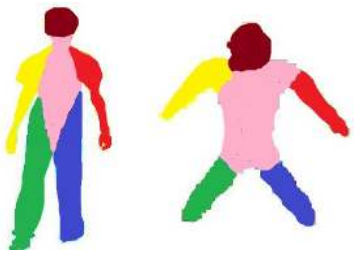
In order to handle local affine changes, Gopalan et al. [38] proposed a shape-decomposition technique that divides a shape into convex parts using Normalized Cuts [39]. These parts are then individually affine normalized and combined into a single shape that is matched using IDSC. As a result, this method is able to capture more deformations of local portions, such as a 3D part articulation that may be modeled by a 2D affine transformation of its projection. This yields a significant improvement over IDSC in many cases. It nevertheless assumes an *a-priori* shape decomposition from a single shape that may be inconsistent in the presence of occlusions or noise in shape extraction. Furthermore, the matching is still global and hence one will be unable to handle partial occlusions of the shapes.

In this work, we propose a locally deformable matching technique that does not require one to make an *a priori* assumption about the decomposition of a shape contour. Rather, the contour decomposition is determined during matching by an efficient search for the decompositions of two contours (into Groups-of-Segments (GSs)) that yield the best matching. The technique not only handles articulations, but also models occlusions and extraneous segments explicitly by skipping non-matching segments during matching. Furthermore, each such Group-of-Segments (GSs) is affine-corrected before matching which uses a robust contour matching technique that handles deformations well. As a result, our method is robust in the presence of various challenges such as intra-class variations, articulations, deformations, partial occlusions and errors in the shape extraction process. This is illustrated by results that show significant improvement over the state-of-the-art, especially in the case of partial occlusions and errors in shape extraction.

The remainder of the paper is organized as follows: Section 2 describes the processes of shape representation and the extraction of possible GSs from shapes. The cost function for shape similarity given a particular GS correspondence across shapes is described in Section 3. Section 4 describes the process of efficiently determining the best GS correspondence that minimizes this cost function using Dynamic Programming. Finally, in Section 5, we show some promising results obtained by our method.

2. Shape Representation

The first task for any shape matching technique is to come up with a representation of shapes such that they can be matched efficiently and accurately. In our problem, the inputs are assumed to be outer shape contours,



(a) (b)

Figure 2: Best viewed in color. Part-decomposition obtained from Gopalan et al. [38] using a single shape above.

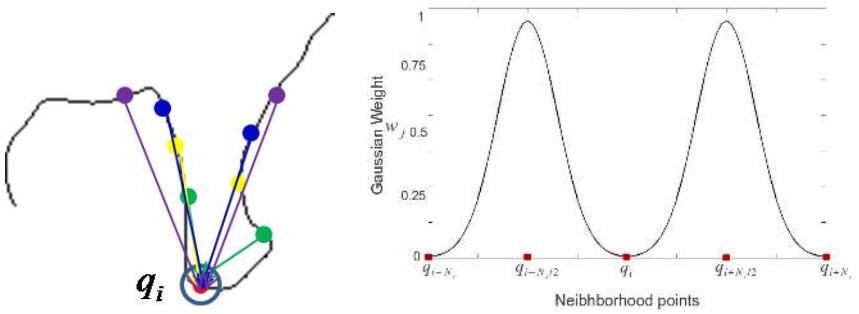
that may be obtained from automatic techniques such as background subtraction and Image Segmentation or in some cases, they may be manually drawn.

Shape Contours have often been represented by decomposing them into small parts so that local transformations can be determined for each part. While decomposition of a shape is important, it is, however, a very challenging task, especially under occlusions or noise. For example, results of shape-decomposition using a single shape using the method proposed by Gopalan et al. [38] are shown in Figure 2 and it may be inferred that the method more or less fails in consistently segmenting a shape into the same parts in different shape instances. This lead to errors when the individual parts across such shapes are attempted to be matched. To deal with this problem, in this work, we consider multiple decomposition possibilities in our shape representation and chose the shape decomposition pair that matches best across two given shapes.

First, we break the whole shape contour into small straight line-like segments. Then, Groups of such Segments (GSs) are created. Then, the shape decompositions that are allowed for a given shape are taken to be collections of such GSs such that they don't overlap, with some skips/unassigned segments allowed in the shape decomposition. We next discuss how to extract possible *break-points* from a shape contour that will form the possible start and end points of GSs. This is done in order to restrict the number of points at which the GSs can start and end.

2.1. Computing Possible Break Points for GSs

Typically, points at which a local geometric transformation changes are coincident with points of high curvature. Hence, the points of high curva-



(a) (b)

Figure 3: (a) Some of the possible neighborhood points around the contour point q_i to calculate *Angle Sharpness*. Pairs of the neighborhood points are shown in the same color. (b) The Gaussian weighting function.

ture are determined first. Several approaches [40, 41, 42, 43, 44] have been proposed in the literature for high curvature point detection on contours. In this work, we try to detect an optimal number of points that are distributed across the entire contour and are also robust to possible noise in the contour. To this end, we first calculate *Angle Sharpness* S_a at a contour point q_i using NL (Neighborhood point list) which contains N_s points on either side of q_i. *Angle Sharpness* S_a for a contour point q_i is defined as:

$$S_a(q_i) = \sum_{(q_{i-j},q_{i+j})\in NL_{q_i}} w_j \cdot (180 - A(q_{i-j}, q_i, q_{i+j})), \tag{1}$$

where $A(q_{i-j}, q_i, q_{i+j})$ is the angle between the contour points q_{i-j}, q_i and q_{i+j}. To calculate the curvature at a certain scale which depends on N_s, we introduce a scheme whereby two Gaussians centered at $q_{i+(N_s/2)}$ and $q_{i-(N_s/2)}$ respectively are used. The Gaussian weighting function makes the whole procedure quite robust to noise since the result depends on many points and not on a single point alone. Figure 3 shows an example of such a computation.

Candidate curvature points are identified by considering local maxima of the *Angle Sharpness* above a certain threshold. Furthermore, lower maxima close to a higher maximum are removed as they provide more or less duplicate information. We call the points thus detected as *high curvature points*.

We further notice that all possible break points between local groups of segments cannot be modeled using only high curvature points as GS junctions. Figure 4 shows points (in green) across which articulation occurs and

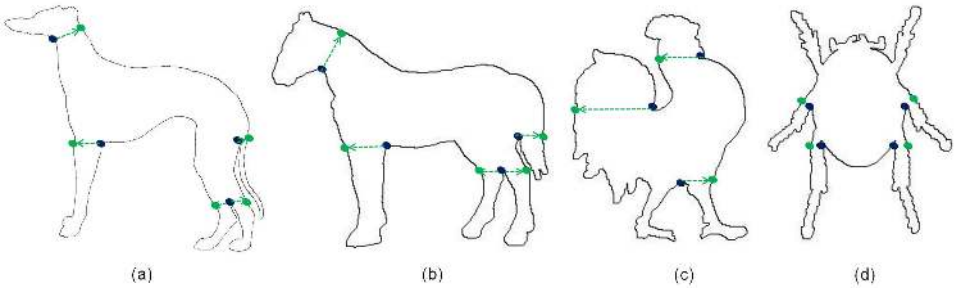
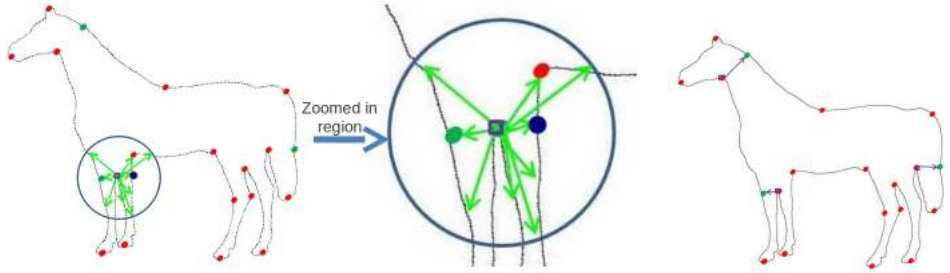


Figure 4: Possible non-convex articulated points (*opposite points*) extracted by our algorithm.

the local transformation of the object shape often changes. Thus, we detect additional points known as *opposite points* in this work.

To detect an *opposite point*, we first determine *concave points* based on the complexity measure proposed by Gopalan et al. [38]. It has been observed in several prior works ([45, 46, 47, 38]) that each part of a shape is typically convex. Any two points within a convex region have the same Euclidean Distance (ED) and Inner Distance (ID) while a concave region has different ID and ED where the inner-distance (ID)[37] is defined as the length of the shortest path within the shape boundary. The shortest path is a collection of line segments and the intermediate vertice(s) on such shortest paths between points lying in two different convex regions represent the *concave points*. Although these *concave points* typically coincide with high curvature points detected in our approach, they help in extracting *opposite points* since they are at the joint of two convex parts. Figure 5 shows such a *concave point* with a square.

For a given *concave point* p, we consider all the contour points at Geodesic distance less than or equal to a distance d as candidate *opposite points*. This distance d is fixed as 20 percent of the total number of contour points in the contour. The candidate *opposite point* c, for which the distance between p and c is a local minimum and the ED and geodesic distance (GD) is sufficiently large, is considered as the *opposite point* for *concave point* p. The green point in Figure 5(a) is such an *opposite point*. If there is any high curvature point in a close neighborhood of c, then c represents the same information as that point and it is therefore taken to be the *opposite point* instead of point c. This is again demonstrated in Figure 5(a) using a blue *opposite point* and red *high curvature point*. The procedure for extracting the



(a) (b)

Figure 5: (a) An example of computing the *opposite point* for a given *concave point*. (b) *Opposite points* determined for a horse shape contour.

opposite point is very similar to [48]. Both *opposite points* and *high curvature points* are considered as possible *break-points* in this work.

There are cases where either the shapes or its portions are simple and hence, the possible break points in them cannot be determined, especially from a single shape. For e.g. if the hand of a person is straight, it is very hard to determine from a single shape that it can bend at the elbow. Similarly, there is a need for *break-points* at the points of occlusions which cannot be determined *a priori*. Thus, we ensure that at least one breakpoint exists within a certain range of the contour which ensures that the set of possible GSs has enough number of possibilities that can be used to match with their counterparts in the other shapes. Therefore, we also add extra points known as 'max-size points' to our *break-points* to handle the case of insufficient number of *break-points*. These are added in such a way that the geodesic distance d_k between consecutive *break-points* is maintained. It is taken to be a fraction of the total number of points on the contour (value of d_k is 0.1 used in our experiments). The portion between consecutive *break-points* is defined as a *segment*. The process of creating possible GSs using these *break-points* is described next.

2.2. Computing Possible Groups-of-Segments (GSs)

We determine possible GSs by considering portions of the contour between any two *break-points*. A portion $p_{i,j}$ between any two *break-points* i and j is taken to be a possible GS if it satisfies a certain complexity range: $C_{MIN} \leq C(p_{i,j}) \leq C_{MAX}$. The *Complexity* C is defined as the sum of the angles

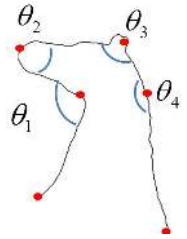


Figure 6: Angles between adjacent segments for an extracted GS. The complexity is the sum of such angles: $\theta_1 + \theta_2 + \theta_3 + \theta_4$.

between consecutive segments constituting the GS.

$$C(p_{i,j}) = \sum_{l=i}^{j-1}(180 - A(Seg_l, Seg_{l+1})) \tag{2}$$

Eq. 2 measures the Complexity of a GS $gs_{i,j}$. $A(Seg_i, Seg_{i+1})$ gives the angle between the segments, Seg_i and Seg_{i+1}, where the angle is calculated from the lines joining the end points of the segments. Figure 6 shows an example of such a computation. A less complex GS is too simple to match and can match with anything whereas considering a highly complex GS leads to rigid matching and a very significant computational expense while matching. Thus, the range of C_{MIN} and C_{MAX} helps in choosing a subset of all GS possibilities that is computationally efficient and is sufficient for most cases. Values of $C_{MIN} = 40$ and $C_{MAX} = 600$ are used for the experiments in this paper. Using the above complexity limit, some GS examples thus extracted: $p_{1,3}$, $p_{1,4}$, $p_{1,5}$ and $p_{1,6}$ for a butterfly shape in Figure 7(a) are shown in Figure 7(b).

2.3. Affine Shape Normalization of each GS

As object and their GSs may appear different in different images due to viewpoint changes or intra-class variations, we perform an affine normalization of each GS. Figure 8 shows an example of shape contours that may look globally different due to intra-class variations, but their affine normalized GSs look quite similar. Non-rigid deformations that may still exist within the affine normalized GSs are handled by contour matching techniques such as the Fast Directional Chamfer Matching as detailed in the next section.

An affine normalization is done using the second order moment matrix

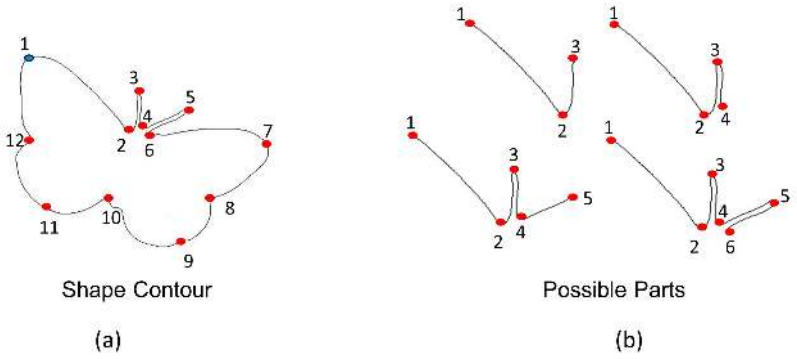


Figure 7: (a) Possible *break-points* on a butterfly contour. (b) Some possible GSs obtained for (a).

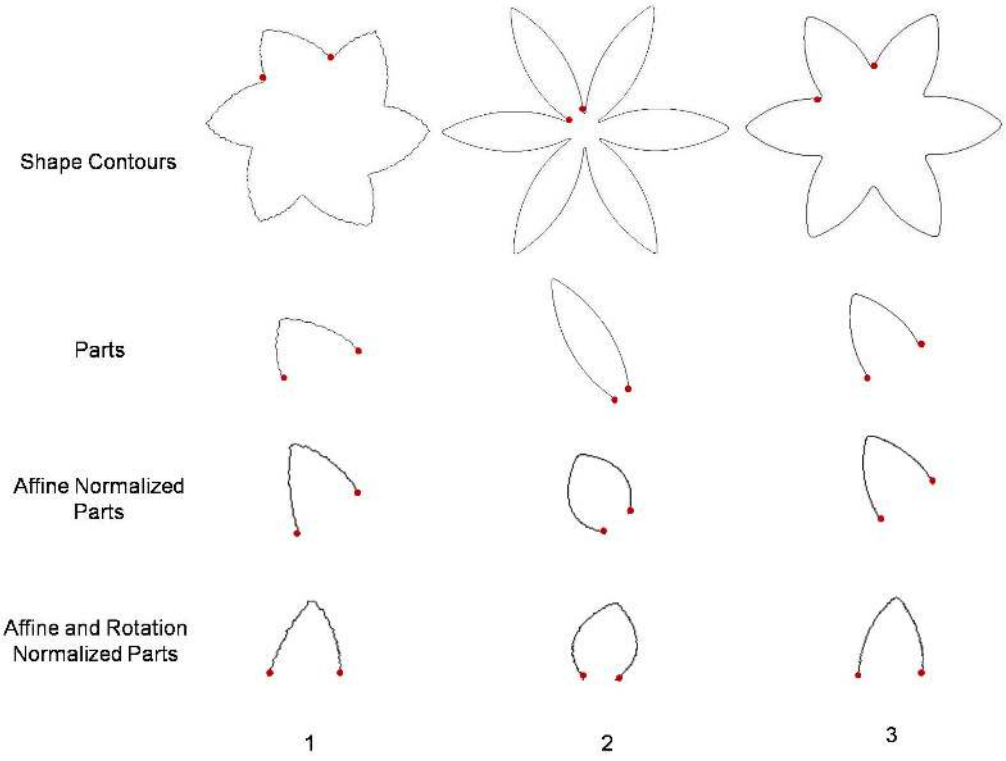


Figure 8: Affine normalization of GSs of some example shapes for a device-1 category of MPEG-7 dataset [19].

M of the contour points:

$$\mathrm{M} = \begin{bmatrix} \sum (x-\bar{x})^2 & \sum (x-\bar{x})(y-\bar{y}) \\ \sum (y-\bar{y})(x-\bar{x}) & \sum (y-\bar{y})^2 \end{bmatrix} = \sum \mathbf{x}_\mu \mathbf{x}_\mu^T$$

By making both the eigenvalues equal, one can estimate the affine-normalization matrix in a manner similar to Cohignac et al. [49] using :

$$\boxed{\mathbf{A} = \mathbf{M}^{-1/2}}$$

Such a normalization corrects the affine transformation, but only up to a rotation as it can also be shown that an arbitrary rotation applied to the contour points does not affect the eigenvalues of the Moment Matrix. Hence, rotation normalization is applied to make it rotation invariant, using a technique based on whether the GS contour is open or closed. For an open GS contour, rotation transformation is estimated by aligning the *start* and the *end* points to some fixed points on the horizontal axis. This handles the scale variation as well. On the other hand, for a closed contour, the rotation is estimated by aligning the *start point* of the contour and the *centroid* of the contour to some fixed points on the horizontal axis as the centroid remains the same under an affine transformation.

Such normalization yields locally affine and rotation normalized GSs. Figures 9(a) and 9(b) show some examples of affine shape normalization for open and closed GS contours respectively.

Given such possible GSs for any two shapes, the cost function for matching them is explained next.

3. Shape Similarity

Given possible GSs in Images I_1 and I_2, we define a match between the shapes as a set of GS correspondences across such possible GS sets that satisfy certain constraints. Specifically, there must be *non-overlap*, i.e. two matched GSs in an image should not intersect with each other. Next, the order between GSs must be preserved, i.e. if one GS is before another GS in an image, such an order must be preserved in the other image.

For a given GS match list (ML) that satisfies these constrains, we define the cost function that evaluates the goodness of a match. How to optimize such a cost function to determine the best matching will be discussed in later section (Section 4).

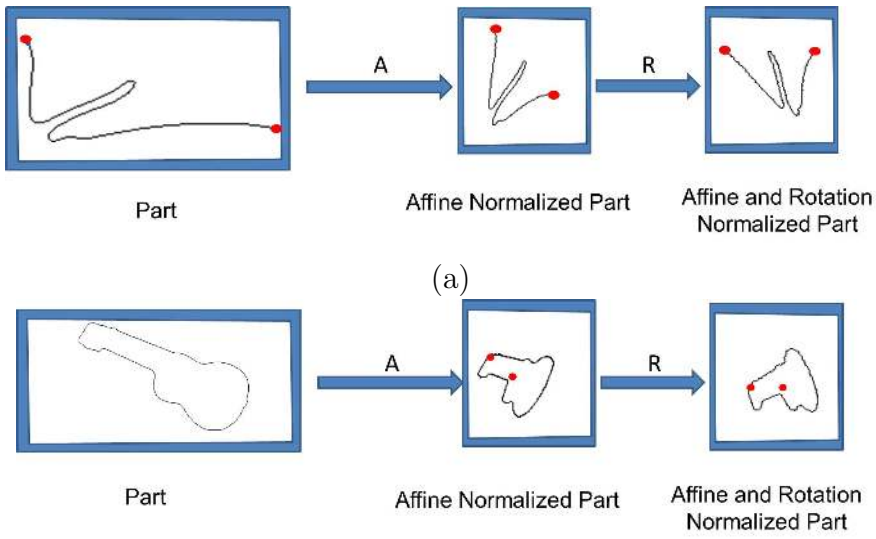


Figure 9: Affine Shape Normalization for (a) an open and (b) a closed GS.

Let the GS match list ML contain GSs $gs^1_{i,j}$ and $gs^2_{l,m}$ of Images I_1 and I_2 respectively and let the entry in ML before $gs^1_{i,j}$ and $gs^2_{l,m}$ be denoted as $gs^1_{prev(i,j)}$ and $gs^2_{prev(l,m)}$, which we refer to as *previous GSs*. The similarity score for ML is defined using three components: *Unary Cost* (C_{uc}), *Binary Cost* (C_{bc}) and the *Skip Cost* (C_{skip}).

$$C(ML) = \sum_{pair(gs^1_{i,j}, gs^2_{l,m}) \in ML} (C_{uc}(gs^1_{i,j}, gs^2_{l,m}) + C_{ipc}(gs^1_{i,j}, gs^2_{l,m}, gs^1_{prev(i,j)}, gs^2_{prev(l,m)})) + C_{skip} \tag{3}$$

3.1. Unary cost

The *Unary cost* (C_{uc}) has two components: the *Matching Cost* (C_{match}) and the *Complexity Cost* ($C_{complex}$). The *Unary Cost* of each GS evaluates how similar and complex these GSs are, considered individually.

$$C_{uc}(gs^1_{i,j}, gs^2_{l,m}) = C_{match}(gs^1_{i,j}, gs^2_{l,m}) + C_{complex}(gs^1_{i,j}, gs^2_{l,m}) \tag{4}$$

The *Matching Cost* (C_{match}) between the GSs measures their relative similarity. In this work, we have used *Fast Directional Chamfer Distance*

(FDCM) proposed by Liu et al. [21] for matching the GS contours which works reasonably well as it captures the edge orientation information better compared to the traditional *Chamfer matching* [50] or the *Oriented Chamfer matching* [51]. Such a matching module can deal with non-rigid deformations or noise present in the affine normalized GSs. In our experiments, we have empirically chosen the number of orientations as 20 for calculating the directional distance transform. To make the matching computationally efficient, we individually precompute the directional distance transform for each GS of a shape contour. Then, the match of another GS with this GS can be computed in an extremely efficient manner.

The average FDCM score between affine normalized versions of $gs^1_{i,j}$ and $gs^2_{l,m}$ is represented as $C_{dc}(gs^1_{i,j}, gs^2_{l,m})$ and the *Matching Cost* C_{match} between them is defined as:

$$C_{match}(gs^1_{i,j}, gs^2_{l,m}) = w_{p_{(i,j),(l,m)}} \cdot C_{dc}(gs^1_{i,j}, gs^2_{l,m}) \tag{5}$$

where

$$w_{p_{(i,j),(l,m)}} = w^1_{i,j} + w^2_{l,m} \quad \text{where} \tag{6}$$

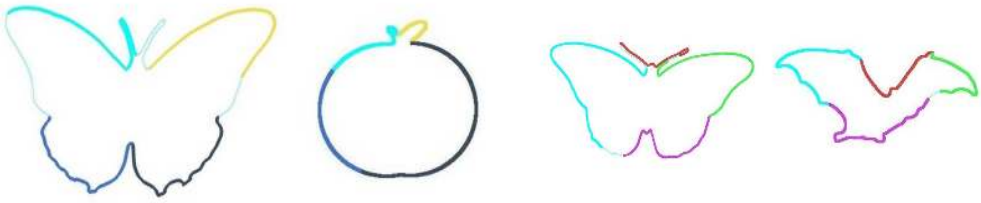
$$w^k_{i,j} = \frac{N(gs^k_{i,j})}{N^k} \tag{7}$$

where $w^1_{i,j}$ and $w^2_{l,m}$ are the weights for the matched GSs $gs^1_{i,j}$ and $gs^2_{l,m}$ in Images I_1 and I_2 respectively and it can be seen that the weights normalize to 1: $w^k_{n,1} + \sum_{j=1}^{n^k-1} w^k_{j,j+1} = 1$. n^k represents the number of *break-points* in Image k, $N(gs^k_{i,j})$ represents the number of contour points in GS $gs^k_{i,j}$ whereas N^k represents the total number of contour points in Image k. Such a weighting scheme helps in bringing different matches to the same scale regardless of the number of contour points in each shape or the number of GSs in a match.

The matching cost alone does not tell the full picture as some GSs are easier to match than other. Hence, we introduce a *Complexity Cost*, where the *complexity* is defined as before. This helps in choosing the GSs that are relatively more complex and discriminative in terms of their shape structures. The *Complexity Cost* between GSs $gs^1_{i,j}$ and $gs^2_{l,m}$ is defined as:

$$C_{complex}(gs^1_{i,j}, gs^2_{l,m}) = w_{gs_{(i,j),(l,m)}} \cdot \left(\frac{\alpha_c}{C(gs^1_{i,j})} + \frac{\alpha_c}{C(gs^2_{l,m})}\right) \tag{8}$$

where $w_{gs_{(i,j),(l,m)}}$ is the weight of the GSs obtained from Eq. 6. $C(gs^k_{i,j})$



(a) (b)

Figure 10: Best viewed in color. Some example shapes look globally quite different but their affine-corrected GSs are similar. Note that non-matching portions are skipped, which are shown with a light blue color.

represents the Complexity measure for a GS $gs_{i,j}^{k}$ (the same as in Section 2.2, Eq. 2). Such a definition defines the complexity of matching a GS much better than simply counting the number of segments in it. The value of α_c that controls the weight given to the *Complexity Cost* is empirically set to 300 in our experiments.

The *Unary Cost* (C_{uc}) as defined in Eq. 4 helps in choosing sufficiently similar, complex GSs between the two images. Generally, the *Unary Cost* is sufficient to distinguish between two shapes under articulations and viewpoint changes. However, there are cases where the individual GSs are similar even though the shapes themselves are globally different. Some cases are shown in Figure 10. It may therefore be useful to consider some constraints between adjacent GSs as too much size variation or too much articulation at the joint points can lead to a significant distortion in the shape. This is considered next, although the weight for these factors is taken to be much less than the unary costs in our standard implementation, although this can be easily varied as per the requirements of a given application.

3.2. Binary Cost

Binary Costs account for Angle and Scale Inconsistencies between adjacent GSs. Enforcing the preservation of these consistencies helps in limiting the possible articulations at the joint points of these GSs that can sometimes change the shape perspective very drastically. The *Binary Cost* does not give full global shape matching perspective like some other methods which enforce global constraints, but does so in a soft way giving a shape a second-level perspective beyond individual GS level which helps in improving the results in most cases. The relative weights of these costs can be varied as per the

shape variations present in a given application. For more deformable shapes, a much lower cost for binary costs compared to the unary costs should be applied to ensure that flexibility while for more rigid shapes, unary costs should be given a higher weight.

We define the *Binary Cost* as:

$$C_{bc}(gs^1_{i,j}, gs^2_{l,m}, gs^1_{prev(i,j)}, gs^2_{prev(l,m)}) = C_s(gs^1_{i,j}, gs^2_{l,m}, gs^1_{prev(i,j)}, gs^2_{prev(l,m)}) + C_a(gs^1_{i,j}, gs^2_{l,m}, gs^1_{prev(i,j)}, gs^2_{prev(l,m)}) \quad (9)$$

Scale Inconsistency C_s at the joint between adjacent GSs is evaluated by measuring the changes in the scale ratios of consecutive GSs. The scale ratio of consecutive GSs is a more appropriate criterion for representing shapes than using the scale of the whole shape for normalization since the latter may be unreliable especially in the presence of occlusions or noise. The *Scale Inconsistency* for two consecutive pairs of GSs is defined as:

$$C_s(gs^1_{i,j}, gs^2_{l,m}, gs^1_{prev(i,j)}, gs^2_{prev(l,m)}) = \alpha_s \cdot w_{ip_{(i,j),(l,m)}} \cdot (1 - e^{-\beta_s \cdot \triangle s_{(i,j),(l,m)}}) \quad (10)$$

where

$$w_{ip_{(i,j),(l,m)}} = w^1_{prev(i,j)} + w^1_{i,j} + w^2_{prev(l,m)} + w^2_{l,m} \quad (11)$$

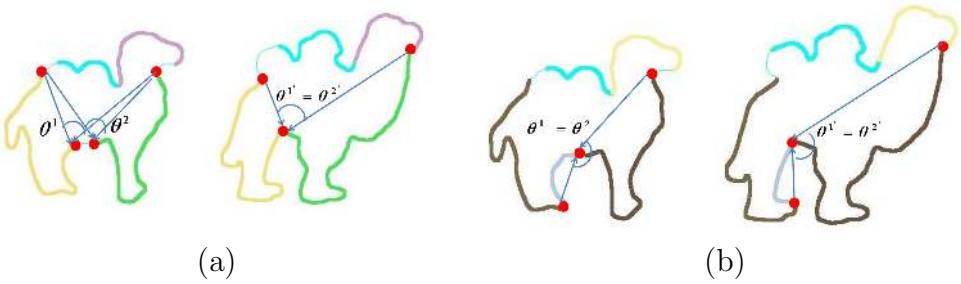
$$\triangle s_{(i,j),(l,m)} = abs(\frac{N(gs^1_{prev(i,j)})}{N(gs^1_{i,j})} - \frac{N(gs^2_{prev(l,m)})}{N(gs^2_{l,m})}) \quad (12)$$

Here, $\triangle s_{(i,j),(l,m)}$ represents the change of relative scale between consecutive GSs: $gs^1_{i,j}$ and $gs^1_{prev(i,j)}$ in I_1 and their corresponding GSs in I_2. Thus, from Eq. 12, we see that large changes in the relative scale of adjacent corresponding GS pairs incur a large penalty. Weights are as defined in Eq. 7.

Similarly, the *Angular Inconsistency* is defined as:

$$C_a(gs^1_{i,j}, gs^2_{l,m}, gs^1_{prev(i,j)}, gs^2_{prev(l,m)}) = \alpha_a \cdot w_{ip_{(i,j),(l,m)}} \cdot (1 - e^{-\beta_a \cdot \Delta\theta_{i,j,l,m}}) \quad (13)$$

where, $\Delta\theta_{i,j,l,m} = \max(|\theta^1_{prev(i,j),(i,j)} - \theta^{1'}_{prev(l,m),(l,m)}|, |\theta^2_{prev(i,j),(i,j)} - \theta^{2'}_{prev(l,m),(l,m)}|)$ represents the change in the relative angle between consecutive GSs $gs^1_{i,j}$ and $gs^1_{prev(i,j)}$ of Image I_1 with respect to the corresponding GSs in Image I_2. The method of measuring $\theta^1_{prev(i,j),(i,j)}$, $\theta^2_{prev(i,j),(i,j)}$ in Image I_1 and their cor-



(a) (b)

Figure 11: Best viewed in color. GS correspondences (a) with scale and angular constraints, (b) without scale and angular constraints. Note that non-matching portions are skipped, which are shown with a light blue color.

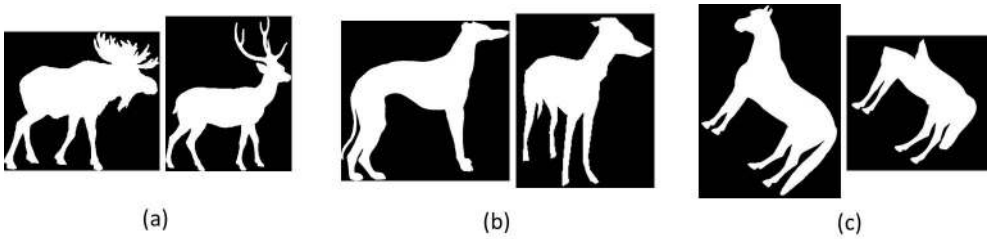


Figure 12: Portions of shapes can be very different in two matching shapes due to (a) intra-class variation, (b) viewpoint change and (c) occlusions and errors in shape extraction.

responding angles $\theta^{1'}_{prev(l,m),(l,m)}$ and $\theta^{2'}_{prev(l,m),(l,m)}$ in Image I_2 is shown in Figure 11. Constant values of $\alpha_a = 200$, $\alpha_s = 200$, $\beta_a = 0.09$ and $\beta_s = 1.5$ are used in the experiments.

3.3. Skip Cost

There are scenarios where some portions of the contour can be missing in either one or both of the images. Such scenarios arise due to self-occlusions and/or viewpoint changes. Furthermore, for certain categories, some portions of the contour look totally different due to intra-class shape variations. Also, shape contours can appear as combinations of multiple other shapes when the input contours are obtained using segmentation or Background Subtraction techniques. These cases are demonstrated in Figure 12. To handle such challenges, *Skip Cost* (C_{skip}) that facilitates partial matching of the shapes is considered.

The *Skip Cost* is a penalty for segments that have been skipped because they do not have a match. It is calculated using the skipped contours in both

the images as:

$$C_{skip} = C^1_{skip} + C^2_{skip} \tag{14}$$

where

$$C^k_{skip} = \sum_{Seg_i \in SSL^k} \beta_{skip} \cdot \omega^k_i \tag{15}$$

where SSL^k(*Skipped Segments List*) is the list that contains the segments in Image k which do not have a match and the weight ω^k_i is as defined before. Value of $\beta_{skip} = 210$ is used in our experiments.

Given the possible GSs in Images I_1 and I_2, the procedure for extracting the best matching list between two given shapes in terms of the energy function is described next.

4. Dynamic Programming-based Matching

The problem of contour matching is defined in terms of determining a Match List that minimizes the energy function. The value of the energy function of a particular configuration depends both on how well the corresponding GSs match and the binary relationships between the consecutively matched GSs. We realize that the entries of ML (in Eq. 3) are circular which makes the problem hard. However, if one neglects the binary constraints between the last and the first GS, one can break the cycle and solve the problem exactly using *Dynamic Programming*(DP). This approach is used in our work due to its much greater efficiency. The problem thus obtained is similar to the *Longest Common Sub sequence* (LCS) [52] problem, where one needs to match a common subsequence of tokens across two given sequences. However, there are some additional considerations due to the *non-overlap* and the *binary* constraints between the matched GSs.

More precisely, let the sets of possible GSs of Images I_1 and I_2 be $\mathcal{GS}_1$ and $\mathcal{GS}_2$ respectively. Then, we require a one-to-one mapping from $\mathcal{GS}_1$ to $\mathcal{GS}_2$ in an *order-preserving* manner such that the matched GSs should not have any overlapping segments between them. Using the cost function defined in the previous section, we define the best matching between two contours as:

$$ML^* = arg\min_{ML} \Big(\sum_{pair(gs^1_{i,j}, gs^2_{l,m}) \in ML} (C_{uc}(gs^1_{i,j}, gs^2_{l,m}) + C_{ipc}(gs^1_{i,j}, gs^2_{l,m}, gs^1_{prev(i,j)}, gs^2_{prev(l,m)}) + C_{skip} \Big) \tag{16}$$

wherfe ML is a match list that contains matched GS correspondences as defined in the previous section. To solve the problem exactly for the cost function defined in Eq. 16, one needs to search over all possible GS combinations. The solution space becomes very large if one considers all possible skips between the GSs while maintaining order constraints. For that, one would need to build a 2D matrix T such that each element $T(gs^1_{i,j}, gs^2_{l,m})$ represents the minimum total cost of matching the two shapes up to the matching GS-pair ($gs^1_{i,j}$,$gs^2_{l,m}$), to compute which, one would have to search over a large set of combinations of the previous matching GS pairs $(gs^1_{prev(i,j)}, gs^2_{prev(l,m)})$ due to the possibility of skipped segments. The running time of such an algorithm would be $M \times N \times S^2$, where M and N are possible number of GSs in Images I_1 and I_2 respectively and S is the maximum skip allowed.

This is quite expensive and impractical and hence, we consider an approximate solution to the problem that is much faster and works reasonably well in practice. The approximation is done by minimizing the cost function at a block level. A block is considered as a set of possible GS correspondences whose ending *break-points* are the same. Only the best match within a block is saved for the next step in the optimization. This approximation helps in reducing the number of possible skips from quadratic to linear. More precisely, we propose a memory-efficient solution where we store only an $m \times n$ matrix T such that each element $T(i, j)$ represents the minimum total cost of matching between Images I_1 and I_2 up to the matches of the i^{th} and the j^{th} segments of Images I_1 and I_2 respectively. Note that since we now work on segments and not GSs, the dimensions m and n of the matrix T are the possible number of *break-points* or segments in Images I_1 and I_2 and not the number of GSs which is much higher.

The minimum total cost of matching up to the entry (i, j) can be calcu-

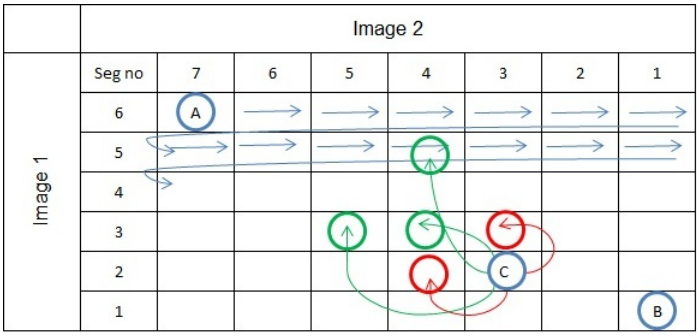


Figure 13: Best viewed in color. Skipping is shown in read circle and some of the possible matches are shown in green color.

lated using Dynamic Programming using the following recurrence relation:

$$T(i,j) = \begin{cases} \infty & if\ i = 0 \| j = 0 \\ min\Big[\min\limits_{gs^1_{i-t,i} \in P^1, gs^2_{j-q,j} \in P^2} \Big[T(i-t, j-q)) + C_{uc}(gs^1_{i-t,i}, gs^2_{j-q,j}) \\ \quad + C_{ipc}(gs^1_{i-t,i}, gs^2_{j-q,j}, lastGS(i-t, j-q, 1), \\ \quad lastGS(i-t, j-q, 2))\Big], \\ \quad T(i-1, j) + skip^1_i, \\ \quad T(i, j-1) + skip^2_j\Big] & otherwise \end{cases} \tag{17}$$

where ($lastGS(i-t, j-q, 1)$,$lastGS(i-t, j-q, 2)$) is the best matched GS correspondence, ending with segment indexes $i-t$ and $j-q$, where t and q represent the number of segments in GSs $gs^1_{i-t,i}$ and $gs^2_{j-q,j}$ of Images I_1 and I_2 respectively. $skip^k_i$ represents the skip cost for skipping the i^{th} segment in Image k. The computation of $T(i,j)$ as described above can be arranged in a sequence as shown in Figure 13, such that all the necessary terms for the calculation of $T(i,j)$ are already available at the time of its computation and is stored in a table along with the least total cost at each stage. We also store the last best matched GS correspondence ($lastGS(i-t, j-q, 1)$,$lastGS(i-t, j-q, 2)$) for each $T(i,j)$, which not only helps in the next stage of the algorithm but also to trace the best GS correspondences in the end, as is typically done in Dynamic Programming solutions [52].

Let N and M be the number of GSs in the Images I_1 and I_2 respectively.

Then, the time complexity of matching two shape contours is $O(N \times M)$ using this algorithm when both the shape contours are already aligned. For handling rotations, one has to consider all possible starting points. Since we extract *break-points* such that the number of starting points in the Image I_1 can be restricted to only the *break-points*, the algorithm is much more efficient compared to other contour point-based approaches ([37, 5]) which have to search over all the contour points for the starting point correspondence. Furthermore, the required memory storage in our approach is $O(m \times n)$, where m and n are the number of segments in Images I_1 and I_2 respectively, which is much less compared to the number of contour points. To give some idea of the actual processing time of our DP-based matching, we ran our code on a 64-bit 2.4Ghz single-core i7 processor machine for matching 100 different preprocessed shape pairs of MPEG-7. Our DP-based matching took 0.355 second to compare two shapes where average number of extracted possible GSs was 126.

In the next section, we compare the performance of our algorithm with other existing approaches on some standard datasets.

5. Experiments

We evaluate our algorithm for shape matching for the task of shape retrieval when the objects are represented only by contours or silhouettes. This is typically the output from many automatic segmentation techniques such as Background Subtraction or Image Segmentation. First, we show results on the popular MPEG-7 shape dataset [19] and compare against other methods that have been considered in the past. Apart from the original dataset, we also show results of matching when the shape extraction has some errors due to missed portions or merged segmentation. This is done by simulating such errors on the MPEG-7 dataset. Then, we evaluate our algorithm against IDSC [37] and Gopalan et al. [38] on a dataset provided by Gopalan et al. [38] created using a real Background Subtraction algorithm on different human and robot poses. The results obtained on such a dataset would be indicative of the performance of the algorithms in scenarios involving real 3D articulated objects. Finally, we show visual results on the 2D Mythological Creatures dataset provided by Bronstein et al. [17] that contains shapes obtained by artificially merging contours of different creatures.

5.1. MPEG-7

The MPEG-7 is a widely used dataset for evaluation of contour-based shape recognition and retrieval methods. It contains 1400 images - 20 shapes per class from 70 different classes. The dataset is challenging because of the presence of deformations, articulations, GS-wise affine changes and missed or altered contour segments in the images.

5.1.1. Results on the Basic MPEG-7 Dataset

Figure 14 shows some matching results depicting also the best matching shape-decompositions obtained by our algorithm on the MPEG-7 dataset. Similar GS color corresponds to a matched GS between the two images. Note that non-matching portions are skipped, which are shown with a light blue color.

Table 1 compares our approach with various existing methods such as [30, 53, 34] for the shape retrieval task on the MPEG-7 dataset. The figures are taken from the respective papers which have reported the Bulleye score for this dataset. The proposed method performs reasonably well as compared to many other techniques. The methods proposed in [54, 55] do not perform individual shape-to-shape matching in isolation, but learn the shape variations present in the dataset in order to improve their performance, taking into account not just the similarity between the shapes of the same category but also the dissimilarity between shapes in different categories in order to train their matching function. Thus, it may be claimed that it is unfair to compare our approach with these learning-based methods that depend heavily on the use of a classified database which may not always be available. Furthermore, it is not clear how the methods would perform on other shapes of a category, such as articulated shapes, if shapes close to those are not present in the database. The method proposed by Gopalan et al. [38] performs the best by affine normalizing each of the convex parts before matching using IDSC.

While the results on the entire dataset are interesting, they provide little insight into the strengths and weaknesses of each method and there is no guarantee that the same ordering of the methods would be obtained on another dataset. A more refined evaluation may be performed by dividing the dataset into 3 categories depending on the type of the shape variations present: the first containing mostly rigid objects, with possible rotations or scale changes; the second containing articulations, deformations and GS-wise affine changes and the last containing missed or altered contour portions. Examples of such categories of shapes are shown in Figure 15.

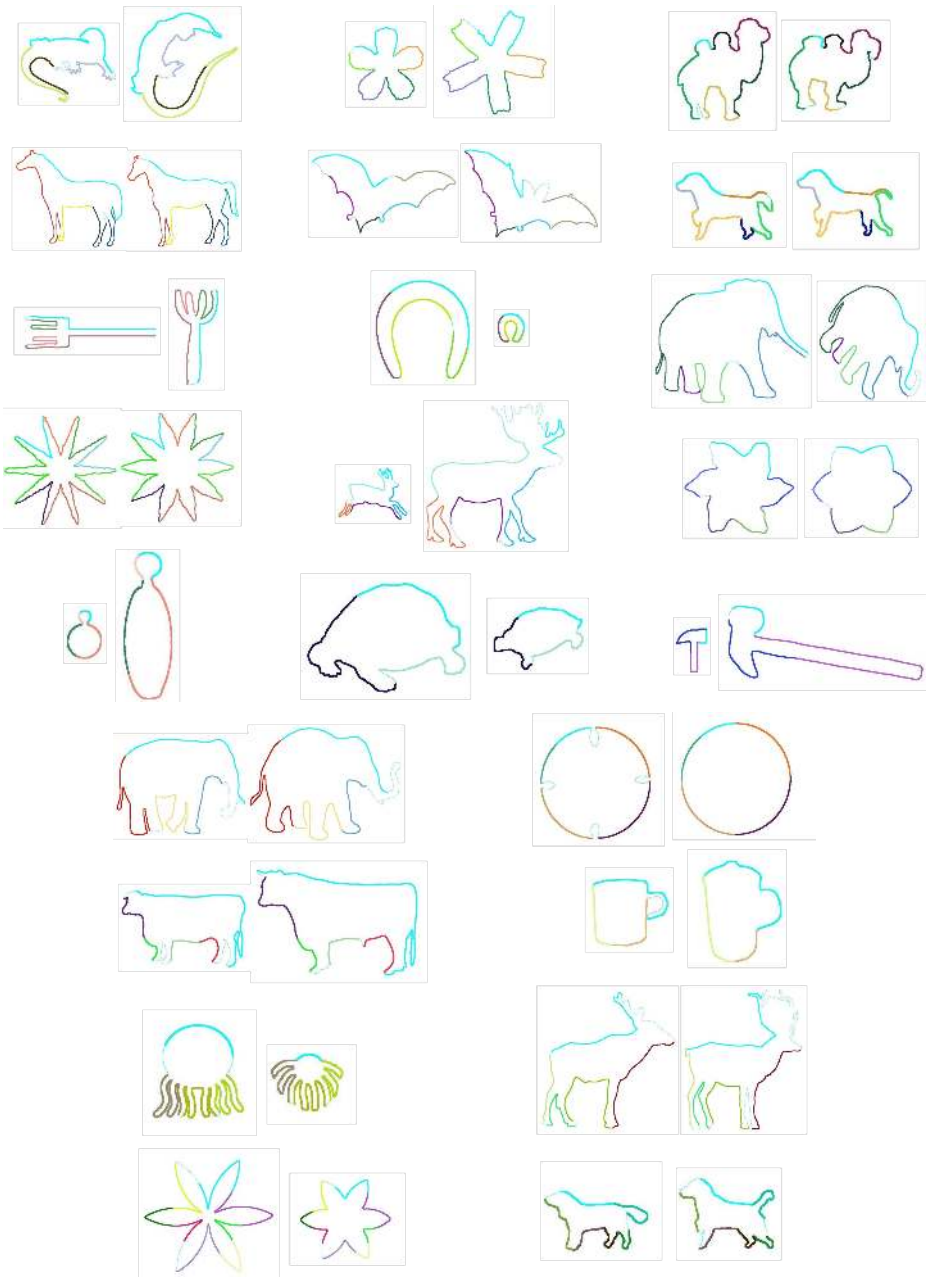
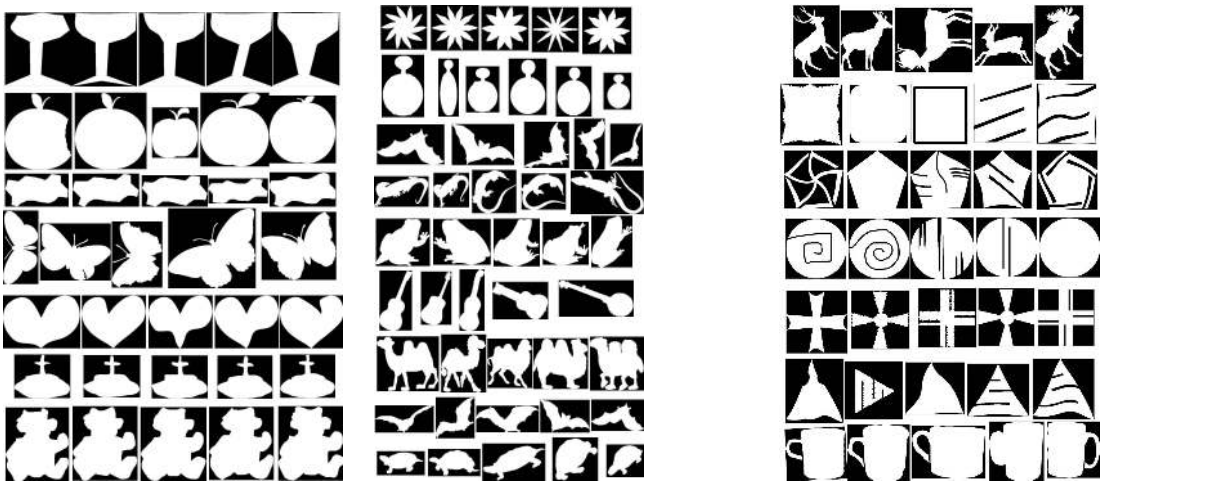


Figure 14: Best viewed in color. Some matchings and best shape decompositions obtained between pairs of shapes.

Table 1: Comparative retrieval results on the entire MPEG-7 dataset [19].

Algorithm	Bullseye Score(in %)
Visual Parts [19]	76.45
SC+TPS [5]	76.51
Curve Edit [56]	78.14
Generative models [57]	80.03
Curvature scale space [58]	81.12
Chance Probability Function [59]	82.69
Fixed Correspondence [60]	84.05
Polygonal Multiresolution [61]	84.33
Multiscale Representation [29]	84.93
Shape L'Âne Rouge [62]	85.25
IDSC + DP [37]	85.40
Symbolic Representation [22]	85.92
Hierarchical Procrustes [32]	86.35
IDSC + DP +EMD [53]	86.86
Triangle area [15]	87.23
Shape-tree [30]	87.70
IDSC + AspectNorm. + StrandRemoval [34]	88.39
Contour flexibility [31]	89.31
Label Propagation [54]	91.61
Locally constrained diffusion [55]	93.32
IDSC + Affine Normalization [38]	93.67
Ours	88.82



(a) Rigid (b) Articulated (c) Missed or altered contour portions.

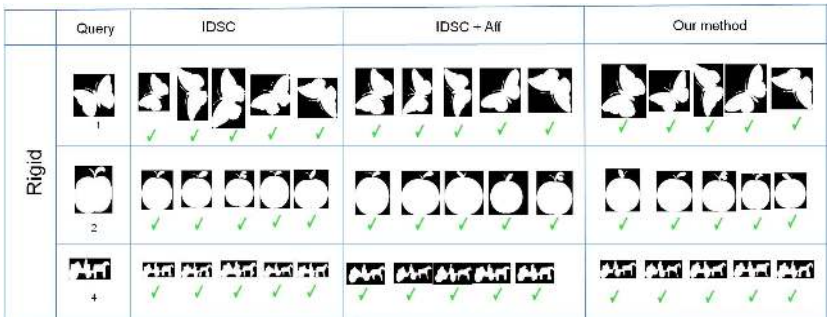
Figure 15: Categorized example shapes from the MPEG-7 dataset.

For these categories, we compare our approach with IDSC and IDSC+Aff [38] for the task of shape retrieval. IDSC is used since the code was freely available[1] and it ran reasonably fast. The implementation of the best reporting one, IDSC+Aff proposed by [38], is not publicly available. Hence, we have re-implemented the paper with the code for shape-decomposition available from the author [2] and tried to reproduce the results as best as possible, testing for different possible parameters. However, we weren't able to reproduce the exact results on the overall dataset as reported by the authors and got slightly lower results (86.9% compared to 93.67%). The category-wise results were not reported in their paper and hence the category-wise comparisons as also the comparisons in the next two sections (MPEG-7 Merged and Partial Occlusion datasets) are done using our re-implementation. However, in spite of such differences in the implementations, the overall pattern of the results should still be the same.

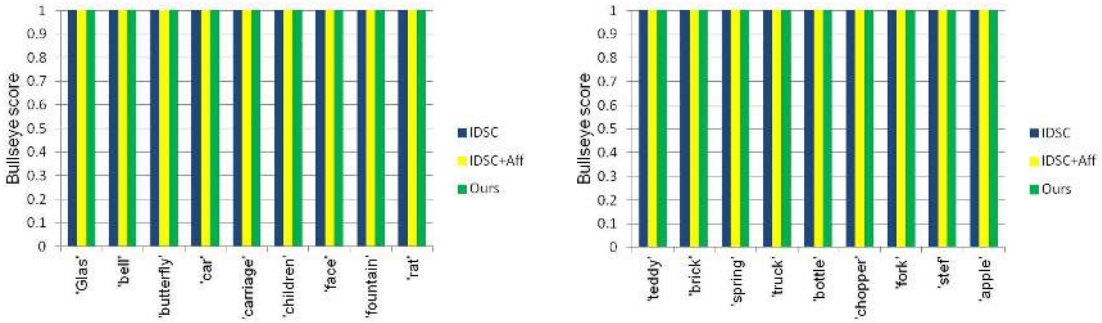
Figure 16(a) visually shows the retrieved results while Figure 16(b) quantitatively compares the category-wise average Bullseye scores for shapes that are mostly rigid but have some deformations. The Bullseye score for a query

[1] *http://www.dabi.temple.edu/~hbling/code/idsc_distribute.zip*

[2] *http://www.umiacs.umd.edu/~raghuram/Segmentation_FULL_NCut.zip*



(a) The 5 most similar shapes retrieved by IDSC, IDSC+Aff and our method.



(b) Shape-wise average Bullseye scores.
Average: IDSC: 100%, IDSC+Aff : 100%, Ours: 100%.

Figure 16: Performance of IDSC vs. IDSC+Aff vs. our approach on the rigid shapes of the basic MPEG-7 dataset which contain little intra-class variations.

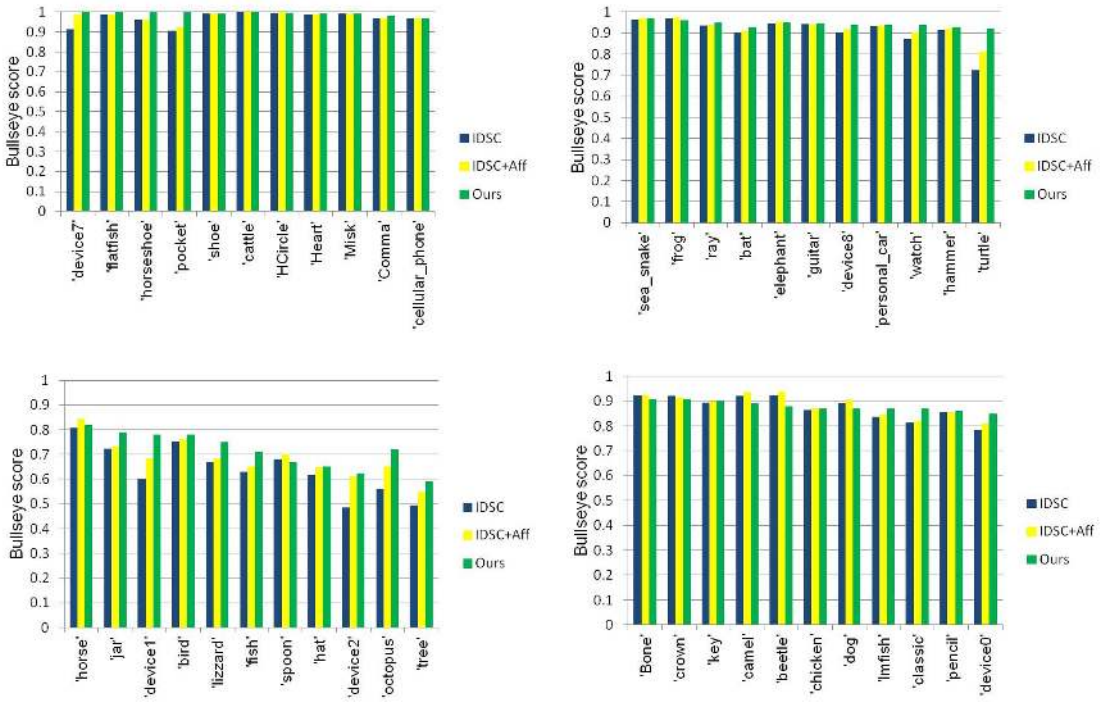
shape is measured by identifying the number of correct retrievals in the top 40 retrieved shapes. As can be seen, all three methods - ours, IDSC and IDSC+Aff - achieve a retrieval rate of 100 percent on all the rigid query shapes. A high performance for this category of shapes for both methods may be attributed to very little intra-class variations. Most of the methods, including those that match shapes rigidly such as the Chamfer Distance ([50]) or the Hausdroff Distance ([20]), should do well on this category of shapes.

Figure 17(a) visually demonstrates the retrieval results, while Figure 17(b) shows the average Bullseye score for the shapes that contain deformations, pose variations, and GS-wise affine changes. It can be seen that IDSC+Aff and our method significantly outperform IDSC by 5 to 20 percent in the average Bullseye score on some of the shapes such as 'turtle','octopus, 'pocket', 'lizard', 'device0', 'device1', 'device2', 'device7', 'classic', 'horseshoe' and 'tree'. The main reason for the improvement seems to be the ability of these algorithms to allow the shapes to undergo different GS-wise affine changes. While articulations can be handled by IDSC itself, it fails to handle these affine variations in the GSs and so scores low on many categories of shapes that have these variations. Our method and IDSC+Aff perform almost the same since both are able to handle such variations. Furthermore, since most of the shapes in the MPEG-7 dataset appear to be in this category, it is not surprising that IDSC+Aff reports good numbers for the entire dataset.

The retrieval performance for shapes with missed or altered contour portions is shown visually in Figure 18(a) and quantitatively in Figure 18(b) where one can note that the improvement in the performance of our method is nearly 10 percent compared to that of both IDSC and IDSC+Aff. Even affine correction of the parts seems to give very little improvement for these categories as the main challenge seems to be the missing portions and hence partial matching of the shapes with skips is required as opposed to a global matching utilized by both IDSC and IDSC+Aff. Examples of such shape alternations include the handle of the 'cup' and the horns of the 'deer' (Fig. 18(a)). Furthermore, global methods may miss important local differences, such as between 'cup' and 'faces', or 'device-9' and 'apples' as shown in Figure 18(a), due to which they confuse between these shapes while more local GS-based matching as is utilized in our work, is able to do much better in such circumstances.

A study of the MPEG-7 dataset shows that 90 percent of the shapes in this dataset belong to the first two categories where global matching meth-

(a) The 5 most similar shapes retrieved by IDSC, IDSC+Aff and our method.



(b) Shape-wise average Bullseye scores.
Average: IDSC: 84.8%, IDSC+Aff : 86.8%, Ours: 88.3%.

Figure 17: Performance of IDSC vs. IDSC+Aff vs. our approach on articulated shapes of the basic MPEG-7 dataset which contains deformations, articulations, and GS-wise affine changes.

ods can perform well, especially if they handle articulations and some GS-wise normalizations and corrections. Thus, it is not surprising that many of these methods ([30, 34, 31]) report good overall performance numbers for this dataset. However, as we have seen for IDSC and IDSC+Aff, their performance is probably not as good on the shapes in the third category, for which more advanced adaptive matching methods are needed. Indeed, the comparative numbers for our algorithm would have been much better if the dataset had more shapes in the third category. The method proposed by Bronstein et al. [17], while being an interesting solution to the problem with good results reported for such cases, is unfortunately computationally extremely expensive and testing on large datasets such as MPEG-7 is prohibitively slow. Thus, our method with much lower running times due to the usage of appropriate approximations and the resulting Dynamic Programming solution seems to be a much better practical solution for handling such more complex shape variations.

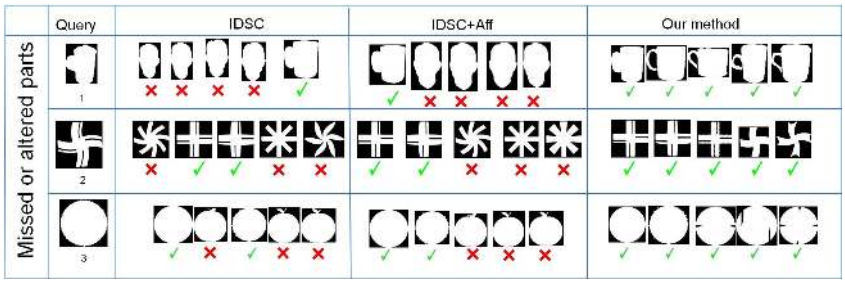
Apart from natural shape variations, automatic shape extraction techniques such as Background Subtraction and Image Segmentation may have errors in the contour extraction process due to which some portions may be occluded/missing, some extra portions may be added or two shapes may merge together. We next simulate the effect of such errors on the MPEG-7 dataset to study the ability of contour matching algorithms to deal with them.

5.1.2. Partially Occluded MPEG-7 Dataset

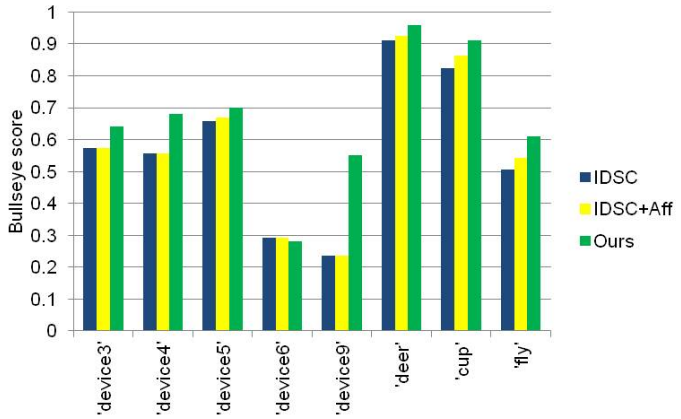
In order to validate the proposed technique in the presence of errors due to occlusions, we modified the standard MPEG-7 dataset by randomly removing n consecutive segments from a shape, where $n = k\%$ of the total number of segments in the shape, k being randomly chosen from $5 - 15$.

Figure 19 shows some visual results of our matching algorithm on this dataset illustrating the shape decompositions, that may be claimed to be quite reasonable since they skip not non-matching portions while matching. Figure 20 visually illustrates the results for the task of shape retrieval for this dataset. Row number 2 presents an interesting instance for the query shape of 'occluded octopus' where IDSC+Aff [38] retrieves only one shape correctly and IDSC retrieves none, while our method results in all but one correct retrievals.

Additionally, we quantitatively evaluate our method on 15 different categories of the Partially Occluded MPEG-7 dataset. We have chosen these



(a) The 5 most similar shapes retrieved by IDSC, IDSC+Aff and our method.



(b) Shape-wise average Bullseye scores.
Average: IDSC: 57.03%, IDSC+Aff: 58.28%, Ours: 66.63%.

Figure 18: Performance of IDSC vs. IDSC+Aff vs. our approach on shapes having missed or altered contour portions in the basic MPEG-7 dataset.

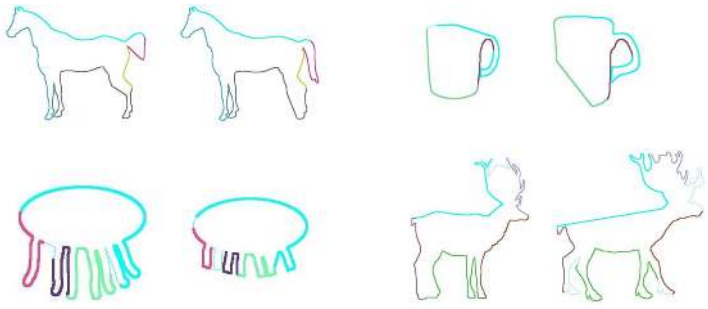


Figure 19: Best viewed in color. Some matching results and shape decompositions obtained between pairs of shapes that have partial occlusions.

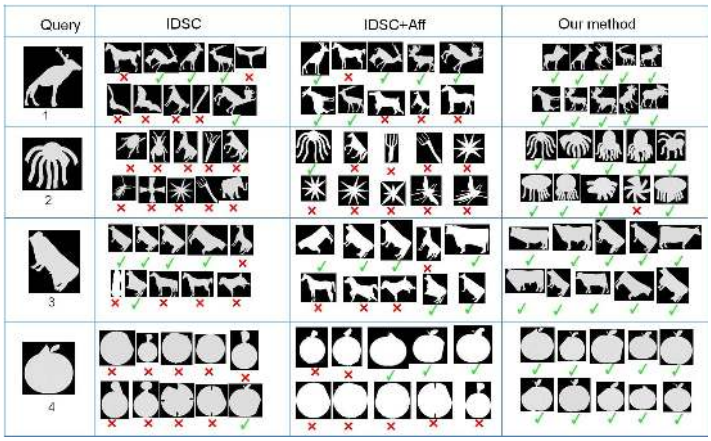


Figure 20: The 10 most similar shapes retrieved by IDSC (second column), by IDSC+Aff (third column) and by our method (four column) for the partially occluded MPEG-7 dataset.

15 categories as IDSC performs reasonably well on these categories in the original MPEG-7 dataset. Figure 21 illustrates the category-wise average Bullseye scores and the overall average scores. The performance of IDSC may be claimed to be quite unsatisfactory as the Inner Distance computation is severely affected by the partial occlusions present in the shapes and the global shape is also quite different. The method proposed by [38] improves the result over IDSC but it does not explicitly model occlusions while matching and has a global approach towards shape similarity. On the other hand, our method performs significantly better due to its ability to model occlusions by skipping certain segments. This way of modeling improves the performance on an average by 38 percent and 14 percent improves over IDSC and the method proposed by [38] respectively.

5.1.3. Merged MPEG-7 Dataset

The other type of error that often arises in the case of contour extraction is the merging of two shapes with each other that can be typically seen as a result of many Background Subtraction or Image Segmentation techniques. In order to simulate this error, we generated 100 different combined shapes by merging two randomly chosen ones. Examples of such shapes are shown in Figure 22.

Figure 23 shows some matchings with their shape decompositions obtained by our algorithm on this dataset where one can see that our method

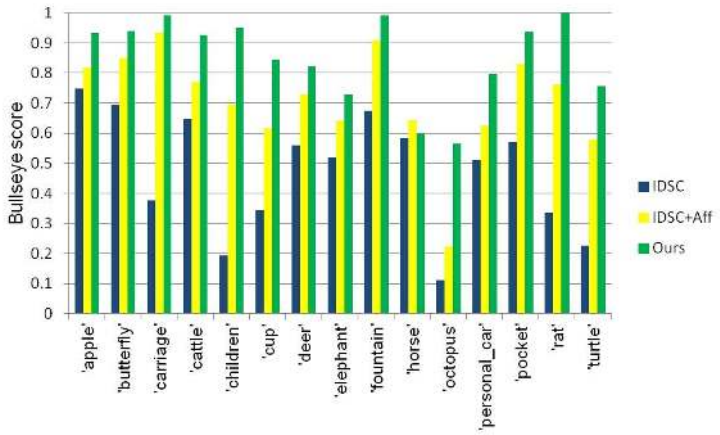


Figure 21: Performance of IDSC vs. IDSC+Aff vs. our approach on the 15 categories of the partially occluded MPEG-7 dataset. Average Bullseye score for IDSC = 47.28 %, IDSC+Aff = 70.92 %, and Ours = 85.2 %.

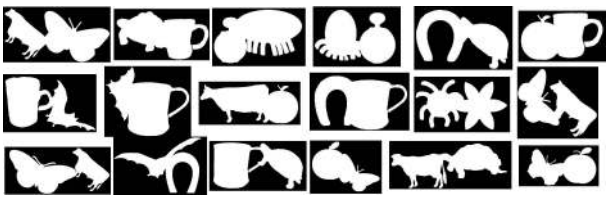


Figure 22: Some examples from our 'Merged MPEG-7' dataset.

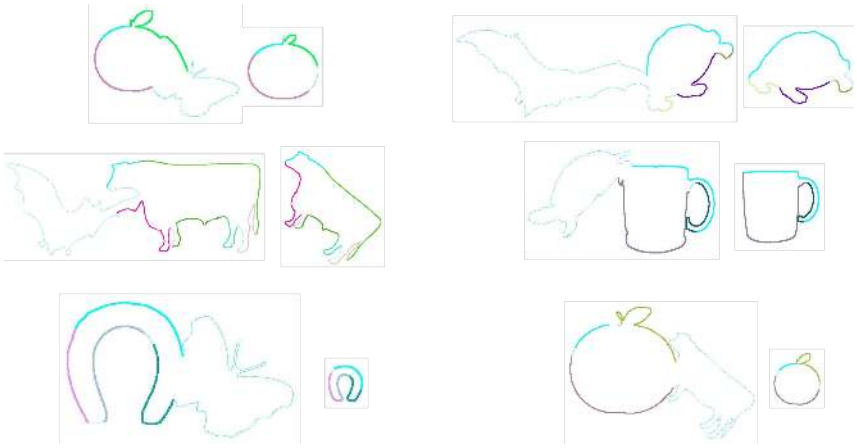


Figure 23: Best viewed in color. Some results of matching and extracted shape decompositions between pairs of shapes when two different shapes merge into a single object shape.

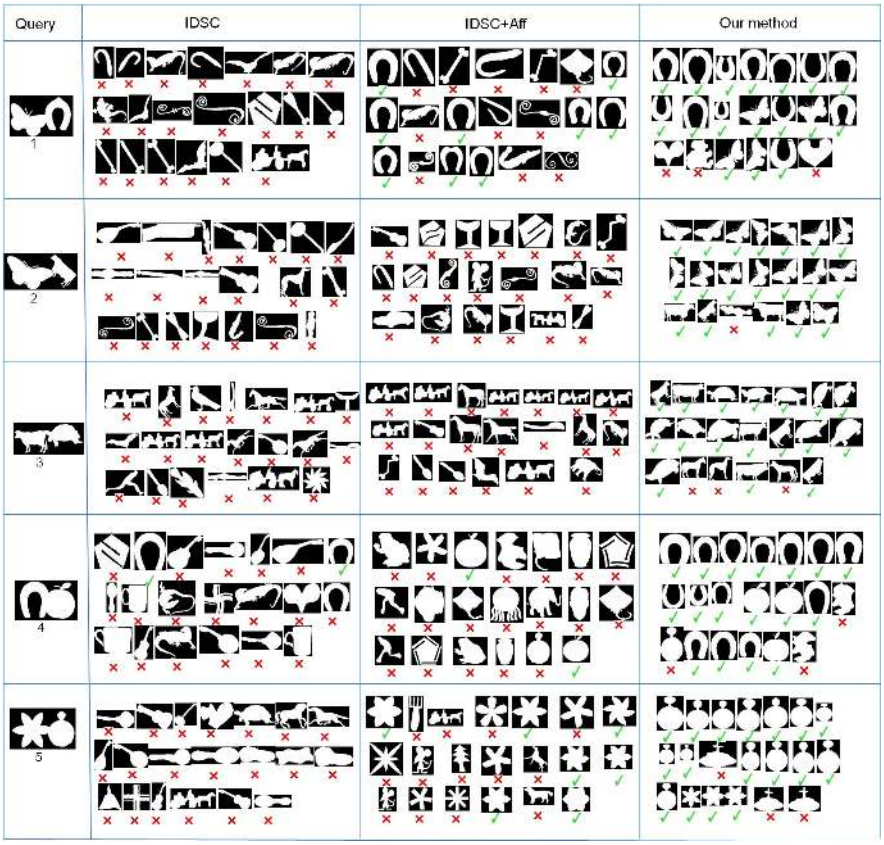


Figure 24: The 20 most similar shapes retrieved by IDSC (second column), by IDSC+Aff (third column) and by our method (four column) for the Merged MPEG-7 dataset.

identifies the correct GS correspondences even if the queries are combined. Figure 24 visually compares the retrieval results obtained by our method, with those of IDSC and IDSC+Aff [38]. Since a query is a combined shape, the retrieved result is classified as a correct match if it contains any of the two shapes present in the query. As can be seen, for most of the queries, while IDSC fails to identify even one correct shape, and the performance of IDSC+Aff [38] is quite unsatisfactory, as our method retrieves mostly correct shapes.

In addition to the visual results, Table 2 compares the recognition rate of our method with IDSC and [38] in a leave-one out environment[37] by comparing the Top-1, Top-5 and Top-10 recognition rates for 100 combined query shapes on the MPEG-7 dataset. IDSC and the method proposed by

Table 2: Comparative retrieval results on 100 different combined shapes from the MPEG-7 dataset.

Method	Top-1 Recognition rate (in %)	Top-5 Recognition rate (in %)	Top-10 Recognition rate (in %)
IDSC [37]	0.19	0.146	0.141
IDSC+Aff [38]	0.27	0.234	0.21
Ours	0.85	0.899	0.843

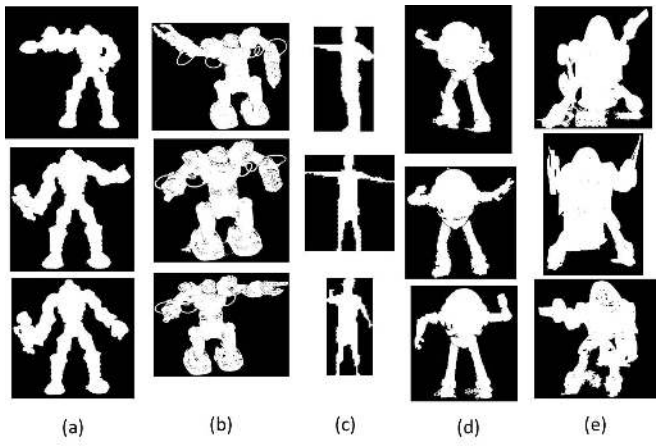


Figure 25: Some example shapes from a real Background Subtraction dataset [38].

[38] can be said to totally fail in this experiment while our method performs reasonably well. This is due to a global criteria for matching and all methods that do not explicitly handle partial matchings should fail miserably on this dataset.

5.2. A Real Background Subtraction Dataset

We next evaluate our algorithm on a real Background Subtraction dataset provided by Gopalan et al. [38]. This dataset contains 50 images of 10 shapes per class from 5 different classes. The dataset has a wide range of non-planar articulations with significant self-occlusions and the images are captured under different viewpoint variations. Many real world scenarios are characterized by such variations. Some examples of such shapes are shown in Figure 25.

Table 3 compares our retrieval results with those of IDSC [37] and Gopalan et al. [38] in a leave-one out environment by listing the Top-1 recognition

Table 3: Comparative retrieval results on a real Background Subtraction dataset.

Method	Top-1 Recognition rate (in %)	Bullseye score (in %)
IDSC [37]	58	39.4
IDSC + Aff [38]	80	63.8
Ours	94	81.5

rate and the Bullseye score. In addition to its shortcomings already discussed, IDSC [37] also fails to capture the 3D articulations of these shapes. Gopalan et al. [38] attempt to do so by performing an affine normalization of GSs. However, many shape variations still remain unmodeled because of the lack of handling occlusions. Thus, it is not surprising that our method significantly outperforms [37] and [38].

5.3. 2D Mythological Creatures dataset

To further test the ability of our algorithm in shape decomposition and matching in the presence of occlusions, we test it on the 2D Mythological Creatures dataset [17]. This dataset contains fifteen shapes of horses, humans and centaurs, that have different articulations and partial occlusions. Figure 26 shows the visual results of our matching algorithm in the presence of articulations, deformations and occlusions. As can be seen, we are able to identify the GS-correspondences across the shapes even in the presence of a lot of distractions. Thus, it may be claimed that our algorithm can be used for such contour correspondence problems.

6. Conclusion

We have introduced an adaptive approach for shape matching that allows for a different affine variation in different portions of a shape. The method does not assume a given shape decomposition *a priori* but determines such decomposition while matching, which makes the matching quite robust. Efficiency is achieved via Dynamic Programming by enforcing an ordering constraint. Further, partial occlusions and errors in contour extraction are handled by allowing skips while matching. Experiments indicate that the method might be useful compared to existing techniques, especially in the case of partial occlusions, extra contour portions and merged shapes that might arise in many situations, including automatic shape extraction using techniques such as Background Subtraction and Image Segmentation.

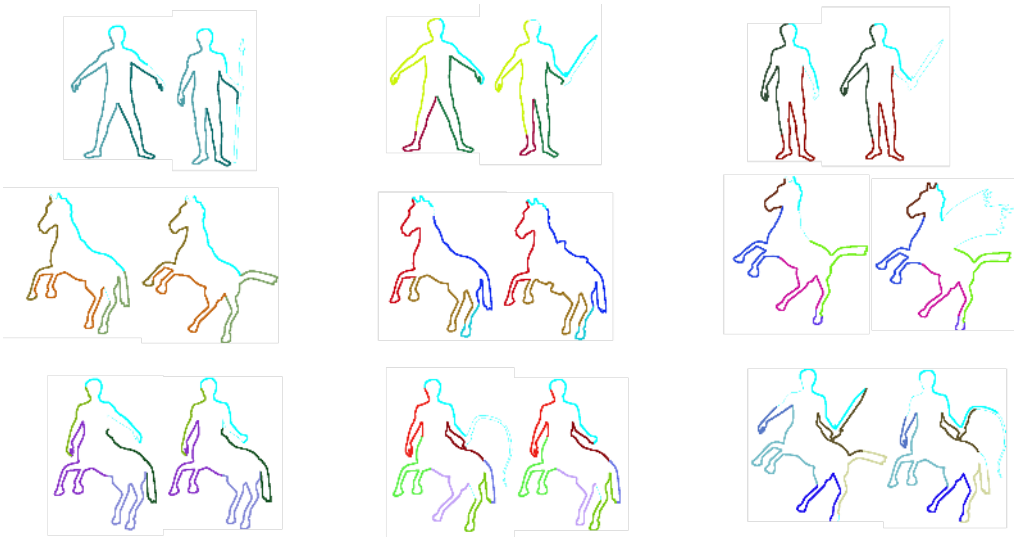


Figure 26: Best viewed in color. Some results of extracted shape decompositions between pairs of shapes on the 2D Mythological Creatures dataset [17]. Note that non-matching portions are skipped.

References

[1] G. Guo, Y. Wang, T. Jiang, A. Yuille, F. Fang, W. Gao, A shape reconstructability measure of object part importance with applications to object detection and localization, International Journal of Computer Vision 108 (3) (2014) 241–258.

[2] X. Wang, X. Bai, T. Ma, W. Liu, L. J. Latecki, Fan shape model for object detection, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2012, pp. 151–158.

[3] C. Lu, L. J. Latecki, N. Adluru, X. Yang, H. Ling, Shape guided contour grouping with particle filters, IEEE International Conference on Computer Vision, 2009, pp. 1–8.

[4] S. Ravishankar, A. Jain, A. Mittal, Multi-stage contour based detection of deformable objects, in: European Conference on Computer Vision, 2008, pp. 483–496.

[5] S. Belongie, J. Puzhicha, J. Malik, Shape matching and object recogni-

tion using shape contexts, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (2002) 509–522.

[6] X. Huang, N. Paragios, D. N. Metaxas, Shape registration in implicit spaces using information theory and free form deformations, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (8) (2006) 1303–1318.

[7] N. Kumar, P. N. Belhumeur, A. Biswas, D. W. Jacobs, W. J. Kress, I. C. Lopez, J. V. B. Soares, Leafsnap: A computer vision system for automatic plant species identification, in: European Conference on Computer Vision, 2012, pp. 502–516.

[8] C. Russell, P. H. S. Torr, P. Kohli, Associative hierarchical crfs for object class image segmentation, in: IEEE International Conference on Computer Vision, 2009.

[9] T. Brox, L. Bourdev, S. Maji, J. Malik, Object segmentation by alignment of poselet activations to image contours, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2011.

[10] D. Zhang, G. Lu, Review of shape representation and description techniques, Pattern Recognition 37 (1) (2004) 1 – 19.

[11] D. Macrini, S. J. Dickinson, D. J. Fleet, K. Siddiqi, Bone graphs: Medial shape parsing and abstraction, Computer Vision and Image Understanding 115 (7) (2011) 1044–1061.

[12] K. Siddiqi, A. Shokoufandeh, S. Dickinson, S. Zucker, Shock graphs and shape matching, International Journal of Computer Vision 35 (1999) 13–32.

[13] T. Sebastian, P. Klein, B. Kimia, Recognition of shapes by editing their shock graphs, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (5) (2004) 550–571.

[14] B. B. Kimia, A. R. Tannenbaum, S. W. Zucker, Shapes, shocks, and deformations i: The components of two-dimensional shape and the reaction-diffusion space, International Journal of Computer Vision 15 (1994) 189–224.

[15] N. Alajlan, M. Kamel, G. Freeman, Geometry-based image retrieval in binary image databases, IEEE Transactions on Pattern Analysis and Machine Intelligence 30 (6) (2008) 1003–1013.

[16] P. Felzenszwalb, Representation and detection of deformable shapes, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (2) (2005) 208 –220.

[17] A. Bronstein, M. Bronstein, A. Bruckstein, R. Kimmel, Analysis of two-dimensional non-rigid shapes, International Journal of Computer Vision 78 (2008) 67–88.

[18] G. Mori, S. Belongie, J. Malik, Efficient shape matching using shape contexts, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (11) (2005) 1832–1837.

[19] L. Latecki, R. Lakamper, T. Eckhardt, Shape descriptors for non-rigid shapes with a single closed contour, in: IEEE International Conference on Computer Vision and Pattern Recognition, Vol. 1, 2000, pp. 424 –429 vol.1.

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

[21] M.-Y. Liu, O. Tuzel, A. Veeraraghavan, R. Chellappa, Fast directional chamfer matching, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2010.

[22] M. R. Daliri, V. Torre, Robust symbolic representation for shape recognition and retrieval, Pattern Recognition 41 (5) (2008) 1782 – 1798.

[23] L. J. Latecki, V. Megalooikonomou, Q. Wang, D. Yu, An elastic shape matching technique, Pattern Recognition 40 (11) (2007) 3069–3080.

[24] Y. Cao, Z. Zhang, I. Czogiel, I. Dryden, S. Wang, 2d nonrigid partial shape matching using mcmc and contour subdivision, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2011, pp. 2345–2352.

[25] R. Basri, L. Costa, D. Geiger, D. Jacobs, Determining the similarity of deformable shapes, Vision Research 38 (1998) 2365–2385.

[26] L. Latecki, R. Lakamper, Shape similarity measure based on correspondence of visual parts, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (10) (2000) 1185–1190.

[27] D. Bryner, E. Klassen, H. Le, A. Srivastava, 2d affine and projective shape analysis, IEEE Transactions on Pattern Analysis and Machine Intelligence 36 (5) (2014) 998–1011.

[28] B.-W. Hong, E. Prados, S. Soatto, L. Vese, Shape representation based on integral kernels: Application to image matching and segmentation, in: IEEE International Conference on Computer Vision and Pattern Recognition, Vol. 1, 2006, pp. 833–840.

[29] T. Adamek, N. E. O'Connor, A multiscale representation method for nonrigid shapes with a single closed contour, IEEE Transactions on Circuits and Systems for Video Technology 14 (5) (2004) 742–753.

[30] P. Felzenszwalb, J. Schwartz, Hierarchical matching of deformable shapes, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2007, pp. 1–8.

[31] C. Xu, J. Liu, X. Tang, 2d shape matching by contour flexibility, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (1) (2009) 180–186.

[32] G. McNeill, S. Vijayakumar, Hierarchical procrustes matching for shape retrieval, in: IEEE International Conference on Computer Vision and Pattern Recognition, Vol. 1, 2006, pp. 885–894.

[33] L. Chen, R. Feris, M. Turk, Efficient partial shape matching using smith-waterman algorithm, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPRW '08., 2008, pp. 1 –6.

[34] A. Temlyakov, B. Munsell, J. Waggoner, S. Wang, Two perceptually motivated strategies for shape classification, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2010, pp. 2289–2296.

[35] T. Ma, L. Latecki, From partial shape matching through local information to robust global shape similarity fo object detection, IEEE International Conference on Computer Vision and Pattern Recognition, 2011.

[36] I. L. Dryden, K. V. Mardia, Statistical shape analysis, John Wiley & Sons, Chichester, 1998.

[37] H. Ling, D. Jacobs, Shape classification using the inner-distance, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (2) (2007) 286–299.

[38] R. Gopalan, P. Turaga, R. Chellappa, Articulation-invariant representation of non-planer shapes, European Conference on Computer Vision, 2010.

[39] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (1997) 888–905.

[40] F. Mokhtarian, R. Suomela, Robust image corner detection through curvature scale space, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (12) (1998) 1376–1381.

[41] X. He, N. H. C. Yung, Curvature scale space corner detector with adaptive threshold and dynamic region of support, in: IEEE International Conference on Pattern Recognition, Vol. 2, 2004, pp. 791–794 Vol.2.

[42] X. Chen He, N. H. C. Yung, Corner detector based on global and local curvature properties, Optical Engineering 47 (5).

[43] T. Tuytelaars, K. Mikolajczyk, Local invariant feature detectors: a survey, Foundations and Trends in Computer Graphics and Vision 3 (3) (2008) 177–280.

[44] H. Liu, L. Latecki, W. Liu, A unified curvature definition for regular, polygonal, and digital planar curves, International Journal of Computer Vision 80 (1) (2008) 104–124.

[45] H. Liu, W. Liu, L. Latecki, Convex shape decomposition, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2010, pp. 97–104.

[46] D. D. Hoffman, W. Richards, Parts of cognition, Cognition 18 (1984) 65–96.

[47] A. Elad, R. Kimmel, On bending invariant signatures fir surfaces, IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (2003) 1285–1295.

[48] Anonymous, Under review, Computer Vision and Image Understanding.

[49] T. Cohignac, C. Lopez, J. M. Morel, Integral and local affine invariant parameter and application to shape recognition, in: International Conference on Pattern Recognition, Vol. 1, 1994, pp. 164–168 vol.1.

[50] G. Borgefors, Distance transformations in arbitrary dimensions, Computer Vision, Graphics, and Image Processing 27 (3) (1984) 321 – 345.

[51] J. Shotton, A. Blake, R. Cipolla, Multiscale categorical object recognition using contour fragments, IEEE Transactions on Pattern Analysis and Machine Intelligence 30 (2008) 1270–1281.

[52] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to algorithms, MIT press, 2001.

[53] H. Ling, K. Okada, An efficient earth mover's distance algorithm for robust histogram comparison, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (5) (2007) 840–853.

[54] X. Bai, X. Yang, L. Latecki, W. Liu, Z. Tu, Learning context-sensitive shape similarity by graph transduction, IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (5) (2010) 861–874.

[55] X. Yang, S. Koknar-Tezel, L. Latecki, Locally constrained diffusion process on locally densified distance spaces with applications to shape retrieval, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2009, pp. 357–364.

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

[57] Z. Tu, A. Yuille, Shape matching and recognition – using generative models and informative features, in: European Conference on Computer Vision, Vol. 3023, 2004, pp. 195–209.

[58] F. Mokhtarian, M. Bober, Curvature Scale Space Representation: Theory, Applications, and MPEG-7 Standardization, 1st Edition, Springer Publishing Company, Incorporated, 2011.

[59] B. J. Super, Learning chance probability functions for shape retrieval or classification, in: IEEE Conference on Computer Vision and Pattern Recognition Workshop, 2004. CVPRW'04., 2004, pp. 93–93.

[60] B. J. Super, Retrieval from shape databases using chance probability functions and fixed correspondence, International Journal of Pattern Recognition and Artificial Intelligence (2006) 1117–1138.

[61] E. Attalla, P. Siy, Robust shape similarity retrieval based on contour segmentation polygonal multiresolution and elastic matching, Pattern Recognition 38 (12) (2005) 2229–2241.

[62] A. Peter, A. Rangarajan, J. Ho, Shape lane rouge: Sliding wavelets for indexing and retrieval, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2008, pp. 1–8.