Efficient Indexing For Articulation Invariant Shape Matching And Retrieval*

Soma Biswas†, Gaurav Aggarwal†† and Rama Chellappa†
Center for Automation Research, UMIACS
†Dept. of ECE, ††Dept. of Computer Science
University of Maryland, College Park
{soma, gaurav, rama}@cfar.umd.edu

Abstract

Most shape matching methods are either fast but too simplistic to give the desired performance or promising as far as performance is concerned but computationally demanding. In this paper, we present a very simple and efficient approach that not only performs almost as good as many state-of-the-art techniques but also scales up to large databases. In the proposed approach, each shape is indexed based on a variety of simple and easily computable features which are invariant to articulations and rigid transformations. The features characterize pairwise geometric relationships between interest points on the shape, thereby providing robustness to the approach. Shapes are retrieved using an efficient scheme which does not involve costly operations like shape-wise alignment or establishing correspondences. Even for a moderate size database of 1000 *shapes, the retrieval process is several times faster than most techniques with similar performance. Extensive experimental results are presented to illustrate the advantages of our approach as compared to the best in the field.*

1. Introduction

Shapes show a great deal of intra-class variability including rotations, translations, articulations, missing portions and other inexplicable deformations which makes the problem of shape matching quite challenging. Many existing shape matching algorithms require computationally demanding matching schemes to account for the variability making them not so desirable for large databases. In contrast, we propose an indexing system for fast and robust matching and retrieval of shapes.

We model a shape as a collection of landmark points arranged in a plane (2D) or in 3D space. In our framework, each shape is characterized by features that are used to index it to a table. The table is analogous to the inverted page table used to index web pages using words/phrases. Given a test shape, similar ones from a pre-indexed collection are determined based on its characterizing features.

As we deal with shapes, the only information available is the underlying geometry. Features are chosen to encode this geometry as richly as possibly, without compromising on robustness. A given shape is represented using a collection of feature vectors, each characterizing a geometrical relationship between a pair of landmark points. Given two points, the following geometrical characteristics are encoded in the corresponding feature vector

1. The inner distance [11] between the points (as opposed to Euclidean which is not articulation invariant),
2. The relative angles between the line segment joining the two points and tangents to the contour at the points,
3. The contour distance between the points (analogous to geodesic distance in case of 3D shapes),
4. Distances of the points from the *articulation-invariant center of mass*. Articulation-invariant center of mass is analogous to the standard center of mass with the added feature of being invariant to articulations.

Clearly, more suitable features can be easily added to this list to make the representation richer. The feature vectors are suitably quantized for indexing. The fact that feature vectors depend only on a few points and are coarsely quantized, provides the necessary robustness required to generalize across large intra-class variability. As shown by the results, the matching speed and ability to generalize does not come at the cost of discriminability across shapes.

The richness and robustness provided by the representation allows the proposed system to have a very simple and efficient retrieval scheme. Given a test shape, the matching bins in the index table are determined. A single parse through the matching bins returns the most similar shapes. This does not require any alignment or correspondences making it extremely fast and scalable.

*Partially funded by the U.S. Government VACE program

1-4244-1180-7/07/$25.00 ©2007 IEEE

The rest of the paper is organized as follows. Section 2 discusses some of the related works. Section 3 introduces the indexing framework proposed in the paper. Section 4 describes the indexable shape representation. A detailed description of the indexing and retrieval algorithms is given in Section 5. Section 6 presents the results of extensive evaluations done to compare the proposed algorithm with others. The paper concludes with a summary and discussion.

2. Previous Work

The indexing approach used in the paper is inspired by the work on fingerprint indexing using minutiae triangles as features [5]. Unlike the classical geometrical hashing [8], the triangle-based approach hashes a set of points based on local invariants (depends only on three minutiae, though need not be local spatially), which is more robust and leads to faster retrieval. For fast matching and retrieval of images, a vocabulary tree based representation is recently proposed by Nister and Stewenius [16]. Similar to their approach, our indexing system relies on invariant and robust shape representation, to make the retrieval process extremely fast. Osada *et al.* [18] use shape distributions for fast matching of 3D models. The idea of describing 3D models using distance between pairs of points and/or their mutual orientations has also appeared in [17] [6] [7] [12].

Shape context based matching has been the theme of several recent works [1] [15] [25] [24] [14] on shape matching. In the classical version [1], each point is characterized by the spatial distribution of the other points relative to it. Similarity computation involves establishing correspondences using bipartite graph matching and thin plate spline (TPS) based alignment. The shape context framework has since been extended by including 1) statistics of tangent vectors [15]; 2) figural continuity constraint [24]; and 3) softassign [2] in shape context framework [25], to suit different requirements of the shape matching problem. One of its recent extensions by Ling and Jacobs [11] accounts for movement of part structures, by replacing the Euclidean distance in the classical version by the inner distance, which is robust to articulations. In addition, the approach involves a dynamic programming (DP) based matching algorithm which helps it to outperform most previous methods. Mori *et al.* [14] show how a shape context-based pruning approach can be used for fast retrieval of similar shapes.

There is another body of work for capturing part structures in which shapes are represented using shock graphs [23] [22]. The shock graph representation is based on the singularities of curve evolution process. Sebastian *et al.* [20] propose a shock graph based method to handle shape deformations. They find the optimal deformation path of shock graphs which brings the two graphs (shapes) into correspondence.

3. Indexing Framework - A Glance

Our focus here is to come up with a fast and efficient framework for shape indexing and retrieval that performs robust shape matching. In most approaches, given a query, it needs to be compared with every shape in the dataset to return the most similar ones. Comparisons often involve computationally demanding operations like registration, establishing correspondence, etc., which are repeated for each shape in the dataset. Such approaches are not scalable and the computational load can become prohibitively high to be useful even for moderate size datasets.

In contrast, we propose a scalable and efficient shape matching and retrieval scheme. Figure 1 illustrates a prototype of our indexing framework. Here, a shape is represented using a set of indexable feature vectors which are appropriately mapped to a *hash table*. For a shape s_k, a bin i in the hash table stores an entry $\langle s_k, n_{ki} \rangle$, $n_{ki} > 0$ where n_{ki} is the number of feature vectors from shape s_k that get hashed to bin i. The hash table is populated by performing the operation for each shape in the database. The resulting table typically has several 2-tuples from different shapes in each bin. The quantization scheme determines how uniformly the entries are distributed across the hash table.

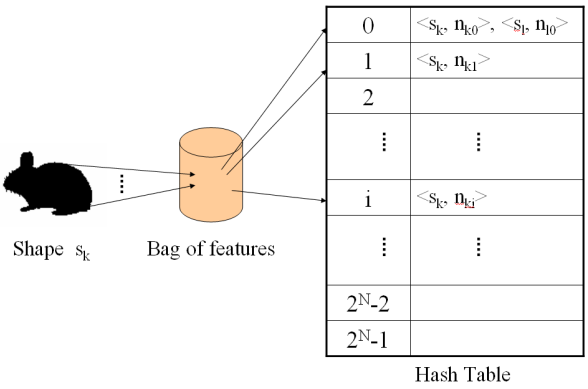

Figure 1. A prototype of the proposed shape indexing framework.

Given a test shape s_t, its feature vectors are extracted and its hash table entries $\langle s_t, n_{ti} \rangle$, $\forall n_{ti} > 0$ are determined by mapping the feature vectors to the table. Once this is done, its similarity with the shapes in the database can be estimated using a single parse through the matching bins. Parsing through the bins that contain a 2-tuple $\langle s_t, n_{ti} \rangle$, one can accumulate the similarity of the query simultaneously with all the shapes in the database. In such a retrieval scheme, the processing time depends only on the number of 2-tuples $\langle s_t, n_{ti} \rangle$ and the number of database entries in the matching bins. Quite clearly, the more uniformly distributed the hash table is, the less is the average time required to process a query. Typically, the processing time increases much slowly as compared to the database size. The details of the

algorithm are described later in Section 5.

4. Shape Representation

Given the indexing scheme, we need suitable features that integrate seamlessly with the framework. The features should not only be indexable but also invariant to deformations like rigid transformations and articulations. This ensures that the single pass retrieval algorithm can directly be used to return the most similar shapes. The choice of features affects both the generalizability and discriminability of the approach. Therefore, we look for features that depend only on a few points on the shape and also take the global shape into account. The dependence on only a few points ensures robustness while their relative configuration with respect to the global shape provides discriminability.

4.1. Pairwise Geometrical Features

Following these guidelines, each shape is characterized by a set of feature vectors where each vector encodes pairwise geometrical relationship between a pair of points on the shape. Each vector consists of the following features that are robust to rigid transformations and articulations.

4.1.1 Inner Distance

The Euclidean distance between two points is invariant to rigid transformations but even small articulations can change the distance for a subset of point-pairs on the shape. Therefore, we use the inner distance (ID) [11] which is robust to articulations of part structures. The inner distance between two points is the length of the shortest path within the silhouette of the shape. Figure 2 illustrates the advantage of inner distance over the standard Euclidean one.

Computation of inner distance involves forming a graph with landmark points on the shape forming the nodes. Two nodes in this graph are connected if there is a straight line path between the corresponding points which is completely inside the shape contour. The corresponding edge weight is the Euclidean distance between the two. From this graph, any standard shortest path algorithm can be used to compute the inner-distance for all the unconnected nodes.

It is worthwhile to note that as desired, the inner distance encodes the global shape to a certain extent, without being overly sensitive to global deformations.

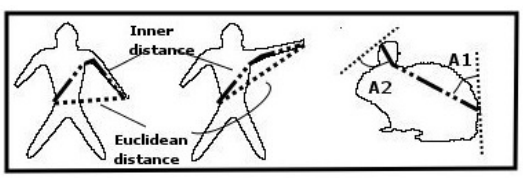

Figure 2. Inner distance and Relative angles.

4.1.2 Relative Angles

Relative angles (A1 and A2) encode the angular relationship between a pair of points. Absolute orientation of the line segment connecting the points is not invariant to rotations. Therefore, relative orientation of the connecting line segment with respect to the incident tangents at each end point is used. When using the inner distance, this becomes the relative orientation of the first segment of the path corresponding to the inner distance (Figure 2). The angles can be computed easily during inner distance computation. Like inner distance, relative angles do take the global shape into account without compromising on robustness.

4.1.3 Contour Distance

The contour distance (CD) is analogous to geodesic distance for 3D shapes. For 2D silhouettes, the contour distance between two points is simply the length of the contour between the two points. It captures the relative positions of the two points with respect to the entire shape contour.

The distance is robust to both articulations and contour length preserving deformations. It complements inner distance in characterizing the relative location of the point pair with respect to the entire shape. Figure 3 shows the contour distance between two points of an object across several deformations. Though the contour distance may seem sensitive to missing points and outliers, we observe that quantization during indexing phase makes it reasonably robust.

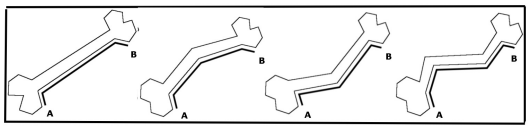

Figure 3. Insensitivity of contour distance to length-preserving deformations.

4.1.4 Articulation-invariant Center of Mass

The features described so far depend on the entire shape, but none of them captures much information about the relative placement of various point pairs in the shape. Though robust, such a representation may not be able to provide desired level of discriminability. To encode the relative placement, one can use the distance of the points and the line segment joining them from the center of mass as additional features. Clearly, these features are not invariant to articulations as the center of mass can change appreciably with articulations. Therefore, we propose an articulation-insensitive alternative to the traditional center of mass.

Here, we first describe how the location of articulation invariant center of mass is determined followed by the features derived from it. Determining such a point directly is

not easy. The proposed approach first transforms a given shape to an *articulation-invariant space*. All objects related by articulations of their part structures get transformed to the same shape in the new space. This essentially means that the distances between the transformed points are invariant to articulations. In other words, the Euclidean distances between transformed points should be the same as the inner distances in the original space.

The transformation is done using multi-dimensional scaling (MDS) [3]. MDS essentially places the points in a new Euclidean space such that the inter-point distances are as close as possible to the given inner distances in a collective manner. We use the classical MDS as opposed to other more accurate but iterative algorithms for efficiency. The transformation computation involves spectral decomposition of inner product matrix B, which is related to the (squared) inner-distance matrix $D_{n\times n}$ as follows

$$\begin{aligned} B &= -\frac{1}{2}JDJ \\ J &= I - \frac{1}{n}\mathbf{1}\mathbf{1}^T \\ \mathbf{1}_{1\times n} &= [1, 1, \cdots, 1]^T \end{aligned} \quad (1)$$

The matrix B is symmetric, positive semidefinite and can be expressed as

$$\begin{aligned} B &= V\Lambda V^T \\ \Lambda &= \mathrm{diag}(\lambda_1, \lambda_2, ..., \lambda_n) \end{aligned} \quad (2)$$

The required transformed coordinates in m dimensional output space can be obtained by

$$X_{n\times m} = V_{n\times m}\Lambda^{\frac{1}{2}}_{m\times m} \quad (3)$$

Figure 4 shows the result of performing MDS on a few shapes. Here m is taken to be two for visualization. The approximation improves with the dimensionality of the output space. As expected, the transformed shapes look quite similar across articulations. The desired articulation-invariant center of mass is the center of mass of the transformed shape. Unlike in the original space, the center of mass of the transformed shapes are almost coincident.

Given the articulation-invariant center of mass of a shape, we derive features which capture the relative positioning of the point pairs. For each point pair, distances (D1, D2, D3) of the points and the line segment joining them from the estimated center of mass are computed. This is done in the transformed space itself as the distances in the transformed space are insensitive to articulations.

4.2. Bag of Features

Given a shape, the pairwise geometrical features are computed for each pair of landmark points on the shape.

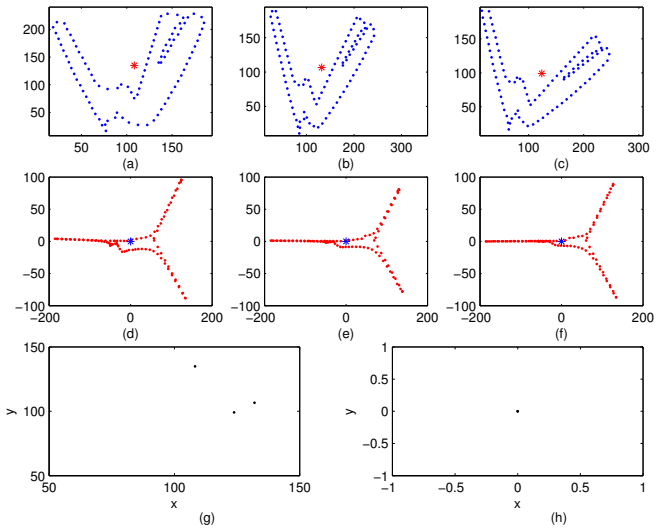

Figure 4. Articulation-invariant center of mass. Row 1: Original shapes, Row 2: Transformed shapes after MDS, Row 3: (left plot) Center of mass distribution of the original shapes, (right plot) Center of mass distribution of the transformed shapes.

Here, each point pair is characterized by a 7 dimensional feature vector, comprising of the features described above. More suitable features can be added for richer representation. The distance based features in the vector are made robust to variations in scale by normalizing each with their medians. The collection of such feature vectors for all pairs of landmark points characterize the shape.

It is worthwhile to note that though the various features are not entirely uncorrelated, they capture different characteristics of the shape. Even experiments show that each one of them contributes to the good performance of the system.

5. Indexing and Retrieval of Shapes

In this section, we describe how the proposed representation is used for shape indexing and retrieval. A shape is indexed by hashing each of its feature vectors to the index table. This requires discretization of the space of feature vectors. Here, we quantize each dimension of the vector independently using a suitably chosen number of levels for each. Suppose $\{f_1, f_2, \ldots, f_7\}$ denotes the 7 dimensional feature vector. The number of levels assigned to each feature are empirically chosen based on the robustness of the feature. If the number of quantization levels for feature f_i is given by 2^{N_i}, then N_i bits are required to represent the feature. So each feature vector consisting of 7 features is represented using $N = N_1 + \cdots + N_7$ number of bits. There are 2^N possible combinations of the feature vectors and hence any vector belongs to one of the $0, 1, 2, \ldots, (2^N - 1)$ bins in the hash table. Table 1 shows the number of bits assigned to each feature in our system.

Table 1. Number of quantization bits for the used features.

ID	A1	A2	CD	D1	D2	D3
4	2	2	4	2	2	2

The quantization boundaries for each feature are chosen such that there are almost same number of entries in each level. This is done by using a set of training shapes which are representative of the database. In addition to being the basic requirement of an indexing system, quantization provides robustness to the variations in the actual values of the features across different instances of the same shape.

5.1. Indexing

Figure 1 illustrates the overall indexing procedure. The steps in the indexing are described below in detail.

1. For each shape in the database, landmark points are extracted from the shape contour. Though one can choose these points judicially, we simply pick points uniformly on the shape contours in all our experiments.

2. For each pair of landmark points, features are computed as described in Section 4. This results in a collection of feature vectors for each shape. If there are n landmark points, we have $\binom{n}{2}$ feature vectors.

3. Each feature vector is quantized using the proposed quantization scheme.

4. The quantized feature vectors are mapped on to the appropriate bins in the hash table. The i^{th} bin contains 2-tuples of the form $\langle s_k, n_{ki} \rangle \, \forall n_{ki} > 0$, where s_k is the k^{th} shape in the database and n_{ki} denotes the number of feature vectors of shape s_k that hash to bin i.

5.2. Retrieval

Given a query shape, the aim is to retrieve the similar shapes in the database as efficiently as possible. Figure 5 illustrates the retrieval phase using a flow chart. The different steps involved in the retrieval phase are enumerated below.

1. Feature vectors for the query shape s_t are extracted in a manner similar to the one used for indexing.

2. Each vector is quantized using the same quantization steps as used for the shapes enrolled in the database.

3. Hashing each feature vector to the index table results in a list of matching bins $M = \{i | n_{ti} > 0\}$, where n_{ti} is the number of query feature vectors which hash to bin i. In general, the number of matching bins is much less than the total number of bins in the hash table.

4. The distance $D(t,k)$ of the query s_t with each shape s_k in the database is initialized to zero.

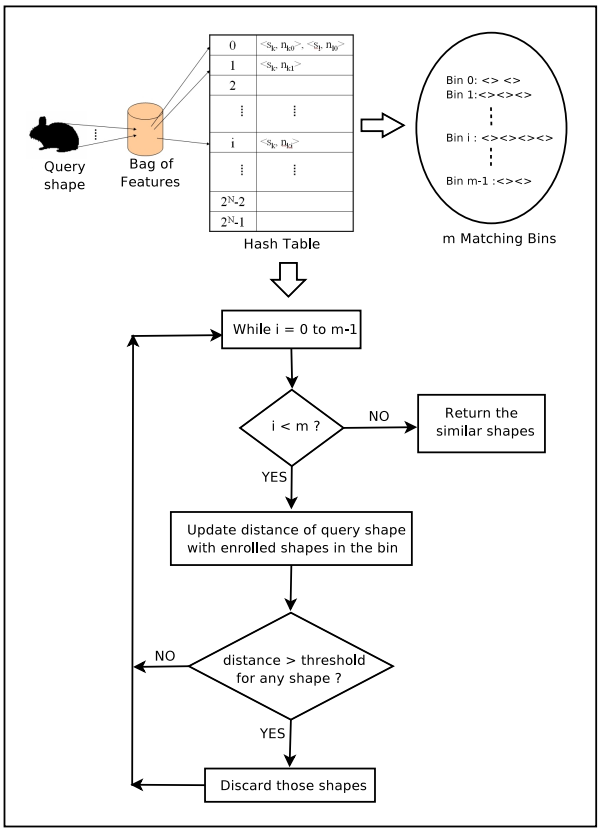

Figure 5. Retrieval Algorithm.

5. Now we parse through the list M and update the distance of the query with each enrolled shape at every step using the following distance metric

$$D(t,k) = D(t,k) + \frac{1}{2}\frac{(n_{ti} - n_{ki})^2}{n_{ti} + n_{ki}} \qquad (4)$$

where the shape s_k has an entry $\langle s_k, n_{ki} \rangle$ in the i^{th} matching bin. If there is no such entry for a shape s_p in the bin, n_{pi} is taken to be zero. The choice of distance metric is inspired by the standard χ^2 statistic.

6. If during parsing, the distance for any particular shape in the database exceeds a pre-specified threshold, then that shape is discarded from further computation.

7. At the end of the parse, we get a list of shapes from the database which are most similar to the query shape.

5.3. Computational Complexity

The computational complexity of the indexing phase depends on the complexity of feature extraction. For a shape with n landmarks, the inner distance computation is of complexity $O(n^3)$. Computation of relative angles and contour distances take $O(n)$. Calculation of articulation invariant

center of mass is $O(n^2)$ while deriving features based on it take $O(n)$. Therefore, indexing a shape takes $O(n^3)$. Note that indexing can be done off-line so that query processing time is not affected. For fairness, all running times reported in the paper include the time spent in indexing.

As in the indexing phase, for a query shape with n landmarks, feature extraction and hashing is $O(n^3)$. Hashing results in $m \ll \binom{n}{2}$ matching bins. Suppose each bin has $p \ll N$ entries, where N is the number of shapes in the database, we need to perform $O(pm)$ distance updates (Equation 4). This does not take into account the fact that a lot of shapes are discarded during retrieval which would further reduce the query processing time. It is difficult to put a bound on how large m and p can be. In the worst case, m can be as large as $\binom{n}{2}$ and p as large as N, but that does not happen in practice. With suitable quantization, p increases much slower than N. Moreover, if elimination of dissimilar shapes during retrieval process is taken into account, the complexity of the process depends on the number of those database shapes which are somewhat similar to the query. These attributes make the system quite scalable.

6. Experiments

In this section, we report the results of empirical evaluation of the proposed system. The performance of the system is compared with many state-of-the art matching algorithms on standard datasets. In addition, we highlight the computational advantages of our indexing approach. Just to give an idea, our system requires only a few minutes (including indexing) to process 1000 queries with a database of size 1000. In comparison most existing systems will probably take order(s) of magnitude longer time to do the same task. In all the experiments, we take 100 uniformly sampled points on the shape contour as landmarks.

6.1. MPEG7 Shape Dataset

As our focus is to show the efficiency of the proposed system along with its accuracy, we first test it on the MPEG7 CE-Shape-1 [10] dataset, which is the probably the largest benchmark used for evaluating shape matching algorithms. The dataset consists of 1400 silhouettes with 20 images each for 70 different objects. Figure 6 shows a few images from the dataset. The standard test for this dataset is the Bullseye test. It is a leave-one-out kind of test where 40 most similar shapes are determined for every query shape. The final score is given by the ratio of the number of correct hits to the best possible number of hits (20×1400).

Table 2 compares the performance and computation time of the proposed approach with many algorithms reported in the literature. The proposed approach takes several order of magnitudes less time than other approaches. (The system runs on a regular desktop and is implemented in MATLAB.) The run-times reported for other algorithms are directly taken from the respective references and may vary slightly due to differences in machine configurations. The accuracy achieved shows that the speed does not come at the cost of robustness. Inner distance shape context (IDSC) with Dynamic Programming (DP) based matching is the only one that performs better, but is much slower that the proposed indexing approach. Interestingly, even the IDSC approach performs worse than the proposed scheme when using shape context distance [1] instead of DP.

Figure 6. Example shapes from MPEG7 CE Shape 1 dataset.

Table 2. Performance comparison on MPEG7 dataset. D_{sc}: shape context distance. DP: dynamic programming based matching.

Algorithm	Score	Computation Time
CSS [13]	75.44%	
Visual Parts [9] [10]	76.45%	
Curve Edit[21]	78.17%	$1s \times {}^{1400}C_2$ (50 segments)
Gen. Models[25]	80.03%	$0.2s \times {}^{1400}C_2$
SC + D_{sc} [11]	64.59%	
SC + TPS [1]	76.51%	$0.2s \times {}^{1400}C_2$
IDSC + D_{sc} [11]	68.83%	
IDSC + DP [11]	85.40%	$0.31s \times {}^{1400}C_2$
Proposed	$\mathbf{81.8\%}$	**10 minutes**

6.2. Articulation Database

The features used in our framework were chosen so as to support articulation-invariant matching. Therefore, it is important to evaluate the performance of the system on a dataset which explicitly deals with large articulations. Here we use the articulation dataset introduced in [11] which consists of 8 objects with 5 shapes each (Figure 7).

We use the same test scheme as in [11]. For each shape, 4 most similar shapes are selected and the number of correct hits for ranks 1, 2, 3 and 4 are calculated. Table 3 summarizes the results obtained. The proposed approach competes well with the other approaches. It is noteworthy that unlike other approaches, our scheme does not require any alignment or dynamic programming-based matching for computing similarity with each shape in the dataset.

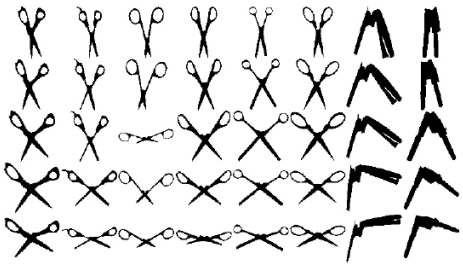

Figure 7. Articulation database

Table 3. Retrieval result on the articulation dataset.

Algorithm	Rank 1	Rank 2	Rank 3	Rank 4
SC + DP [11]	20/40	10/40	11/40	5/40
IDSC + DP [11]	40/40	34/40	35/40	27/40
Proposed	**40/40**	**38/40**	**33/40**	**20/40**

6.3. Kimia Dataset 1 and 2

Kimia dataset 1 [22] consists of 25 shapes from 5 categories. The experiment is run in a leave-one-out pattern. Similar to the articulation dataset, the performance is measured by accumulating the correct matches at ranks 1, 2 and 3. The best one can get at any rank is 25. Table 4 compares the results obtained with other approaches.

Kimia dataset 2 [20] is a larger version of dataset 1. It consists of 99 silhouettes from 9 categories. The performance is measured by examining the correct matches at top 10 ranks for each query. The best one can get for each rank is 99. Table 5 summarizes the results obtained. In addition to being extremely efficient, the proposed approach compares favorably with many existing algorithms.

Table 4. Retrieval result on Kimia 1 dataset.

Algorithm	Rank 1	Rank 2	Rank 3
Sharvit *et al.* [22]	23/25	21/25	20/25
Gdalyahu *et al.* [4]	25/25	21/25	19/25
Belongie *et al.* [1]	25/25	24/25	22/25
IDSC + DP [11]	25/25	24/25	25/25
Proposed	**25/25**	**25/25**	**23/25**

6.4. Gait-based Human Identification

Gait-based human identification techniques use sequences of human silhouettes to characterize a gait. The various silhouettes present in a sequence are essentially the result of articulations of body parts. This motivates us to verify the usefulness of our matching framework for this task. First we perform an experiment to evaluate the ability of our algorithm to handle 3D articulations involved in walking. We take 5 consecutive frames from 10 gait sequences of different individuals from USF dataset [19] (Figure 8) and compute the similarity of each silhouette with every other silhouette. Figure 9 (left) shows the similarity matrix obtained. Darker the similarity matrix is, more similar the two silhouettes are. Ideally the matrix should be 5×5 block diagonal. The result shows the possible usefulness of our approach for the task of gait-based human identification, more so because such an application involves large databases for which one needs a scalable and efficient retrieval scheme. We also perform an identification experiment using the USF data. Each gallery sequence is characterized by indexing its first 10 silhouettes. The similarity of a probe sequence with a gallery sequence is determined by simply combining the similarities of all its silhouettes with the gallery ones. Figure 9 (right) shows the Cumulative Match Curve (CMC) obtained in the experiment with 41 gallery sequences (one per subject) and 41 probes. The performance is quite encouraging even though no temporal information is used in measuring the similarity.

Table 5. Retrieval result on Kimia 2 dataset.

	SC [1]	Gen. [25] Models	Shock Edit [20]	IDSC + DP [11]	**Our Method**
1	97	99	99	99	**99**
2	91	97	99	99	**97**
3	88	99	99	99	**98**
4	85	98	98	98	**96**
5	84	96	98	98	**97**
6	77	96	97	97	**97**
7	75	94	96	97	**96**
8	66	83	95	98	**91**
9	56	75	93	94	**83**
10	37	48	82	79	**75**

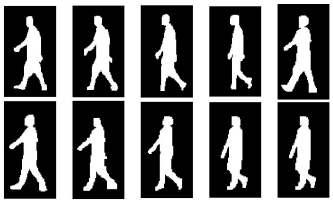

Figure 8. A few silhouettes from the USF gait database. The two rows show silhouettes for two different subjects.

7. Summary and Discussion

We presented an efficient and robust approach for fast matching and retrieval of shapes. The following attributes of the approach contribute towards its robustness and hence graceful degradation of performance in the presence of

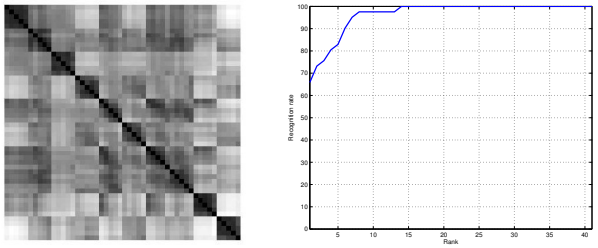

Figure 9. Left: Similarity matrix obtained in matching gait silhouettes. Right: CMC obtained in the gait-based identification.

noise, outliers and other deformations: 1) Pair-wise geometric feature based representation, 2) feature quantization, and 3) invariance of features to rigid transformations and articulations of part structures. Rich and robust feature representation is important even for retrieval process. This helps to achieve robust matching using an extremely simple algorithm not involving any correspondence matching as required by most state-of-the-art techniques. In most existing techniques, the alignment process has to be repeated for every shape in the database for retrieval, making them much slower than the proposed scheme. As dissimilar shapes are eliminated very early during our retrieval process, little effort is wasted in comparing a query to the database shapes which are very different, making the system scalable.

References

[1] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, 2002.

[2] H. Chui and A. Rangarajan. A new point matching algorithm for non-rigid registration. *Computer Vision and Image Understanding*, 89:114–141, 2003.

[3] A. Elad and R. Kimmel. On bending invariant signatures for surfaces. *Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1285–1295, 2003.

[4] Y. Gdalyahu and D. Weinshall. Flexible syntactic matching of curves and its applications to automatic hierarchical classification of silhouettes. *Transactions on Pattern Analysis and Machine Intelligence*, 21(12):1312–1328, 1999.

[5] R. S. Germain, A. Califano, and S. Colville. Fingerprint matching using transformation parameter clustering. *Computational Science and Engineering*, 4(4):42–49, 1997.

[6] A. B. Hamza and H. Krim. Geodesic object representation and recognition. In *DGCI, LNCS 2886*, pages 378–387, 2003.

[7] C. Y. Ip, D. Lapadat, L. Sieger, and W. C. Regli. Using shape distributions to compare solid models. In *ACM symposium on Solid modeling and applications*, pages 273–280, 2002.

[8] Y. Lamdan and H. J. Wolfson. Geometric hashing:a general and efficient model-based recognition scheme. In *Intl. Conf. on Computer Vision*, pages 238–249, 1988.

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

[10] L. J. Latecki, R. Lakamper, and U. Eckhardt. Shape descriptors for non-rigid shapes with a single closed contour. In *Intl. Conf. on Computer Vision and Pattern Recognition*, pages 424–429, 2000.

[11] H. Ling and D. W. Jacobs. Using the inner-distance for classification of articulated shapes. In *Intl. Conf. on Computer Vision and Pattern Recognition*, pages 719–726, 2005.

[12] Y. Liu, H. Zha, and H. Qin. The generalized shape distributions for shape matching and analysis. In *Intl. Conf. on Shape Modeling and Applications*, 2002.

[13] F. Mokhtarian, F. Abbasi, and J. Kittler. Efficient and robust retrieval by shape content through curvature scale space. *Image Databases and Multimedia Search*, pages 51–58, 1997.

[14] G. Mori, S. Belongie, and J. Malik. Shape contexts enable efficient retrieval of similar shapes. In *Intl. Conf. on Computer Vision and Pattern Recognition*, pages 723–730, 2001.

[15] G. Mori and J. Malik. Recognizing objects in adversarial clutter: breaking a visual captcha. In *Intl. Conf. on Computer Vision and Pattern Recognition*, pages 134–141, 2003.

[16] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Intl. Conf. on Computer Vision and Pattern Recognition*, pages 2161– 2168, 2006.

[17] R. Ohbuchi, T. Minamitani, and T. Takei. Shape-similarity search of 3d models by using enhanced shape functions. In *Theory and Practice of Computer Graphics*, pages 97–104, 2003.

[18] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin. Shape distributions. *ACM Transactions on Graphics*, 21(4), 2002.

[19] S. Sarkar, P. J. Phillips, Z. Liu, I. Robledo, P. Grother, and K. W. Bowyer. The human ID gait challenge problem: Data sets, performance, and analysis. *Transactions on Pattern Analysis and Machine Intelligence*, 27(2):162–177, 2005.

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

[21] T. B. Sebastian, P. N. Klien, and B. B. Kimia. On aligning curves. *Transactions on Pattern Analysis and Machine Intelligence*, 25(1):116–125, 2003.

[22] D. Sharvit, J. Chan, H. Tek, and B. B. Kimia. Symmetry-based indexing of image databases. *Journal Visual Comm. and Image Representation*, 9(4):366–380, 1998.

[23] K. Siddiqi, A. Shokoufandeh, S. J. Dickinson, and S. W. Zucker. Shock graphs and shape matching. *Intl. Journal on Computer Vision*, 35(1):13–32, 1999.

[24] A. Thayananthan, B. Stenger, P. H. S. Torr, and R. Cipolla. Shape context and chamfer matching in cluttered scenes. In *Intl. Conf. on Computer Vision and Pattern Recognition*, pages 127–133, 2003.

[25] Z. Tu and A. L. Yuille. Shape matching and recognition: Using generative models and informative features. In *European Conf. on Computer Vision*, pages 195–209, 2004.