Path Similarity Skeleton Graph Matching

Xiang Bai and Longin Jan Latecki, *Member, IEEE*

***Abstract*— This paper proposes a novel graph matching algorithm and applies it to shape recognition based on object silhouettes. The main idea is to match skeleton graphs by comparing the geodesic paths between skeleton endpoints. In contrast to typical tree or graph matching methods, we do not consider the topological graph structure. Our approach is motivated by the fact that visually similar skeleton graphs may have completely different topological structures. The proposed comparison of geodesic paths between endpoints of skeleton graphs yields correct matching results in such cases. The skeletons are pruned by contour partitioning with Discrete Curve Evolution, which implies that the endpoints of skeleton branches correspond to visual parts of the objects. The experimental results demonstrate that our method is able to produce correct results in the presence of articulations, stretching, and contour deformations.**

***Index Terms*— skeleton, skeleton graph, graph matching, shape recognition, geodesic path.**

I. Introduction

Skeleton (or medial axis), which integrates geometrical and topological features of the object, is an important shape descriptor for object recognition [1]. Shape similarity based on skeleton matching usually performs better than contour or other shape descriptors in the presence of partial occlusion and articulation of parts [2][3][4][5]. However, it is a challenging task to automatically recognize objects using their skeletons due to skeleton sensitivity to boundary deformation [6][46]. Usually, the skeleton branches have to be pruned for recognition [6][28][33][34][35][50]. Moreover, another major restriction of recognition methods based on skeleton is a complex structure of obtained tree or graph representations of the skeletons. Graph edit operations are applied to the tree or graph structures, such as merge and cut operations [7][8][9][10][11], in the course of the matching process. Probably the most important challenge for skeleton similarity is the fact that the topological structure of skeleton trees or graphs of similar objects may be completely different. This fact is illustrated in Fig. 1. Although the skeletons of the two horses (a) and (b) are similar, their skeleton graphs (c) and (d) are very different. This example illustrates the difficulties faced by approaches based on graph edit operations in the context of skeleton matching. To match skeleton graphs or skeleton trees like the ones shown in Fig. 1, some nontrivial edit operations (cut, merge, et al.) are inevitable. On the other hand, skeleton graphs of different objects may have the same topology as shown in Fig. 2. The skeletons of the brush in Fig. 2(a) and the pliers in Fig. 2(b) have the same topology as shown in Fig. 2(c).

This paper presents a novel scheme for skeleton-based shape similarity measure. The proposed skeleton graph matching is based on similarity of shortest paths between each pair of endpoints of the pruned skeletons, e.g., see the shortest paths (in red) in Fig. 3. The shortest paths between every pair of skeleton endpoints are represented as sequences of radii of the maximal disks at corresponding skeleton points. We also benefit from the fact that the skeleton endpoints inherit a cyclic order from the contours. This is possible, since the skeletons are pruned based on contour partitioning with discrete curve evolution (DCE) [34], which guarantees that all endpoints of skeleton branches lie on the contour. For example in Fig. 4, all the endpoints (denoted by 1, 2, ..., 6) of the horse's skeleton are vertices of the DCE simplified polygon (in red). The DCE was introduced in [30][31]. An important property of the DCE-based pruning in [34] is its stability in that it is able to remove spurious branches while preserving structurally relevant branches.

The proposed skeleton graph matching method is described in Section 4. In contrast to the existing approaches to skeleton similarity, we do not explicitly consider the topological structure

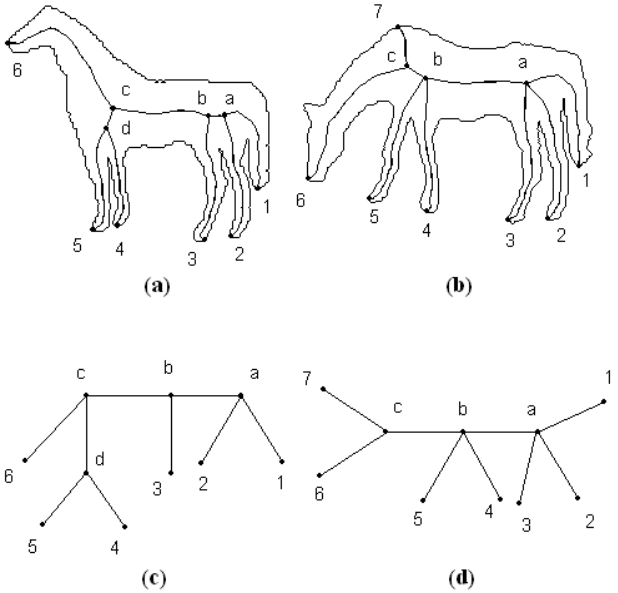


Fig. 1. Visually similar shapes in (a) and (b) have very different skeleton graphs in (c) and (d).

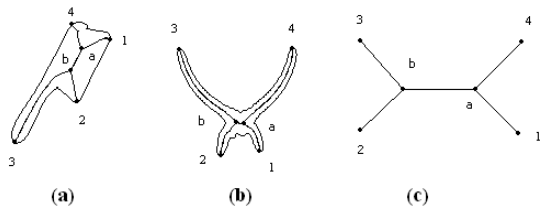
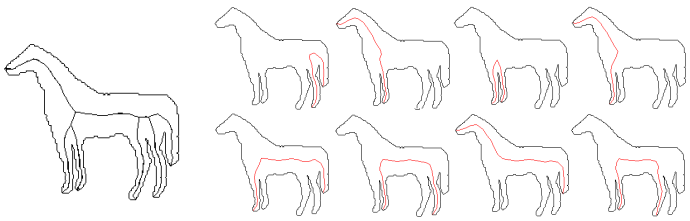


Fig. 2. Dissimilar shapes in (a) and (b) can have the same skeleton graphs (c).

Xiang Bai is with the Dept of Electronics & Information Engineering, Huazhong University of Sci. & Tech. Wuhan, Hubei. 430074 P.R.China, Email:xiang.bai@gmail.com.

Longin Jan Latecki is with CIS Dept.,Temple University, Philadelphia, PA 19094, USA, Email: latecki@temple.edu.



(a) (b)

Fig. 3. (a) The horse's skeleton. (b) The shortest paths (in red) between the pairs of endpoints on the skeleton.

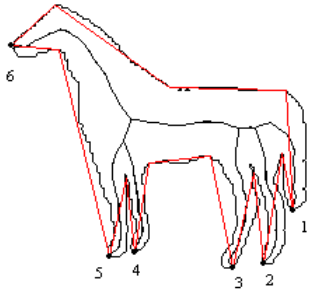


Fig. 4. The skeleton pruned with contour partitioning by DCE [34].

of the skeleton trees or graphs. Instead we focus on the similarity of paths connecting the skeleton endpoints. We use the similarity of the shortest paths between each pair of skeleton endpoints to establish a correspondence relation of the endpoints in different graphs. For example, vertex 1 in Fig. 1(a) corresponds to vertex 1 in Fig. 1(b) since their shortest paths to vertices 2, 3, 4, 5 and 6 are similar. Finally, the dissimilarity value between graphs is easily estimated by the distances between the corresponding endpoints. Thus, the basic idea of our method is to determine the similarity of complex structures (graphs or trees) by examining shortest paths between their endpoints. As we will show in Section 7, the proposed method yields successful recognition results, and is faster than the existing graph and tree matching methods.

The usage of shortest geodesic paths in skeleton graph and in shape similarity is not new; in particular, many-to-many matching in Demirci et al. [27] and the inner-distance in Ling and Jacobs [47] use shortest paths. However, there are substantial differences in our approach. [27] considers shortest paths between all skeleton nodes and [47] considers shortest paths between all contour points. We only consider shortest skeletal paths between skeleton end nodes, which allows us to avoid the problem of the instability of the skeleton junction points (in comparison to [27]) and makes our approach more robust to contour deformations (in comparison to [47]). Moreover, we use skeletal shortest paths in a different and novel way to define shape similarity. In our approach, we use a two-layer structure. In the first layer, skeletal shortest paths emanating from a given skeleton endpoint form its shape descriptor. In the second layer, we compute the similarity of two shapes by matching the shape descriptors of the skeleton endpoints.

Since similar skeletons may have different number of endpoints, we have to allow for a partial correspondence of the endpoints. This is possible with a recently introduced method for partial similarity of sequences [36], which we extend and describe in Section 5. By employing this method, we are able to also match skeletons of object parts to the skeletons of complete objects, and match parts to parts, which is a necessary requirement for robust object recognition.

The proposed skeleton graph matching is based on the assumption that similar skeletons have similar structure of their end nodes (measured by similarity of shortest paths to other end nodes). This assumption is significantly weaker than a standard assumption that a structure of the whole skeleton graph (based on both end nodes and junction nodes) is similar. Usually, the structure of both end nodes and junction nodes is weighted and edited since, as pointed out above (Fig. 1), it is common that skeletons of similar shapes have a different structure of junction nodes. Moreover, as described in Section 2, many approaches to match skeleton graphs require that the graphs are converted to trees prior to finding the correspondence. However, as we will illustrate in Section 7.3, such a conversion may result in lost of important structural information, and consequently negatively influence the object recognition result. The proposed method computes dissimilarity values for graphs that do not have to be trees.

The geodesic skeletal paths are represented as sequences of radii of maximal disks in our approach. Although we do not explicitly consider the topological structure of the skeleton graphs, we do not completely ignore this structure. It is implicitly represented by the fact that overlapping parts of the geodesic skeletal paths are similar, since their overlapping parts have the same subsequences of radii. For our example in Fig. 1(a, b), it means that paths from 6 to 1 and from 5 to 2 overlap. The fact that the overlapping segments are slightly different in (a) and (b) does not affect the similarity of corresponding sequences of radii in (a) and (b). Therefore, our approach is flexible enough to perform extremely well on articulated shapes, but it is not too flexible to confuse dissimilar shapes. This fact is also confirmed by our experimental results in Section 7.

II. Related work

The skeleton-based recognition methods are usually based on the graph or tree representation of the skeletons. Compared with contour matching or other methods, skeleton matching has a lower sensitivity to articulation or rearrangement of parts. However, it involves a higher degree of computational complexity [2][15][37]. Since the skeleton or medial axis is always organized into an ARG (Attributed-Relation Graph), the similarity between two objects can be measured by matching their ARGs. Graph matching is an NPC problem, thus, some efforts have been made to obtain approximate solutions. We review now the most influential solutions proposed in the context of shape similarity.

Zhu et al. match the skeleton graphs of objects using a branch-bounding method that is limited to motionless objects [7]. Liu et al. match ARGs with the A* algorithm. Before matching the axis trees, the merge or cut operation is essential [8][9]. In contrast, the proposed approach does not require any editing of the skeleton graph.

Shock graph is a kind of ARG proposed by Siddiqi et al., which is based on Shock Grammar [16][17][18][19]. Later, Shokoufandeh et al. successfully extend these approaches to structural indexing in a large database [25]. The distance between subgraphs is measured by comparing the eigenvalues of their adjacency matrices. Thus, this method is based on graph topology. It is time consuming because of the complexity of the Shock Grammar and

the calculation of eigenvalues. Sebastian et al. have presented a scheme to compute the edit distance between the shock graphs [14][15], but because of the expensive computation due to the complex operation on shock graphs, the method may fail to deal with occlusion and scene clutter. Torsello et al. use the length of the corresponding boundary segments to edit the similarity of shock trees [23][24].

A different framework is presented in Pelillo et al. [11][12], where hierarchical trees are matched based on finding maximal cliques of the association graphs. Aslan and Tari posit an unconventional approach to shape recognition using unconnected skeletons in the course level [21]. Di Ruberto uses another kind of ARG, called ASG (Attributed Skeleton Graph) [10]. The ASGs are matched with a graduated assignment algorithm, which converts the match matrix (0-1 matrix) into a continuous matrix. This method can deal well with the occlusion problem, however since the matching matrix is obtained using a heuristic rule, the graduated assignment algorithm can find only a suboptimal matching solution.

In recent work, Demirci et al. transform weighted graphs into metric trees for accurate matching [26][27]. However, a heuristic rule is essential to transform graphs to trees (loops need to be removed). An additional problem for this tree matching method is how to select an optimal node as the root, since different root points may have completely different topologies for the same skeleton.

Most of the existing approaches cannot deal with loops. One of few approaches that can deal well with the loop structure is presented in Hilaga et al. [20]. It is a method based on topological matching of Reeb graphs representing 3D models. However, this method distinguishes different shapes only based on topological structure. The proposed method is able to match graphs containing loops, and it yields intuitive results that reflect both geometric and topological shape features.

III. Skeleton graphs

This section describes the initial steps for building the skeleton graphs. The following definitions apply to continuous skeletons as well as to skeletons in digital images (composed of pixels).

***Definition* 1** A skeleton point having only one adjacent point is an endpoint (the skeleton endpoint); a skeleton point having three or more adjacent points is a junction point. If a skeleton point is not an endpoint or a junction point, it is called a connection point. (Here we assume the skeleton curve is one pixel wide.)

***Definition* 2** The sequence of connection points between two directly connected skeleton points is called a skeleton branch. A standard way to build a skeleton graph is as follows: both the endpoints and junction points are chosen as the nodes for the graph and all the skeleton branches between the nodes are the edges between the nodes. For example, Fig. 1(c) and Fig. 1(d) are graphs representing the skeletons in Fig. 1(a) and Fig. 1(b), respectively.

***Definition* 3** The endpoint in the skeleton graph is called an end node, and the junction point in the skeleton graph is called a junction node.

IV. Matching the skeleton-graphs

We match skeleton graphs by establishing a correspondence of their end nodes only, since these nodes are the salient points on the contour and all skeleton branches ending on the contour can be seen as visual parts of the original shape. Thus, the proposed representation does not involve any junction nodes.

A. The shape-path representation

***Definition* 4** The shortest path between a pair of end nodes on a skeleton graph is called a skeleton path, e.g., see Fig. 3(b). Subsection text here.

Suppose there are N end nodes in the skeleton graph G to be matched, and let v_i $(i = 1, 2, \ldots, N)$ denote the ith end node along the shape contour in a clockwise direction. Let $p(v_m, v_n)$ denote the skeleton path from v_m to v_n. We sample $p(v_m, v_n)$ with M equidistant points, which are all skeleton points. Let $R_{m,n}(t)$ denote the radius of the maximal disk at the skeleton point with index t in $p(v_m, v_n)$. A vector of the radii of the maximal disks centered at the M sample points on $p(v_m, v_n)$ is denoted as

$$R_{m,n} = (R_{m,n}(t))_{t=1,2,\ldots,M} = (r_1, r_2, \ldots, r_M). \quad (1)$$

In this paper, the radius $R_{m,n}(t)$ is approximated with the values of the distance transform $DT(t)$ at each skeleton point with index t. Suppose there are N_0 pixels in the original shape S. To make the proposed method invariant to the scale, we normalize $R_{m,n}(t)$ in the following way:

$$R_{m,n} = \frac{DT(t)}{\frac{1}{N_0}\sum_{i=1}^{N_0} DT(s_i)} \quad (2)$$

where $s_i (i = 1, 2, \ldots, N_0)$ varies over all N_0 pixels in the shape.

***Definition* 5** The shape dissimilarity between two skeleton paths is called a path distance. If R and R' denote the vectors of radii of two shape paths $p(u, v)$ and $p(u', v')$ respectively, the path distance is defined as:

$$pd(p(u,v), p(u',v')) = \sum_{i=1}^{M} \frac{(r_i - r'_i)^2}{r_i + r'_i} + \alpha \frac{(l - l')^2}{l + l'}. \quad (3)$$

where l and l' are the lengths of $p(u, v)$ and $p(u', v')$ respectively, and α is the weight factor. In order to make our representation scale invariant, the path lengths are normalized. We include the path lengths in formula (3), since the path length is not reflected in the sequences of radii (all paths are sequences of M radii). This way our path representation and the path distance are scale invariant.

In order to deal with the similarity of articulated shapes, the path distance in (3) does not penalize path deformations (e.g., deformations from straight to curved paths) that do not change the vectors of radii and path lengths. This allows us to recognize as similar two deformable objects such as snakes. It may appear that not penalizing path deformations can lead to a danger of recognizing as similar different shapes. However, while it is possible to deform a given shape (e.g., a snake) so that the vectors of radii and path lengths are constant, it is extremely unlikely to have two different shapes with differently deformed skeletal paths having identical vectors of radii and path lengths. Our excellent experimental results in Section 7 confirm this fact in that we never classified as similar objects of different shapes.

B. Matching end nodes using skeleton paths

In the skeleton graph, each end node has the skeleton paths to all other end nodes in the graph. As we will show the skeleton paths are a useful shape descriptor.

Let G and G' denote two graphs to be matched, and let v_i and v'_j be some end nodes in G and G', respectively. Let the numbers of the end nodes in G and G' be $K+1$ and $N+1$, respectively, and $K \leq N$. The matching cost $c(v_i, v'_j)$ between v_i and v'_j is estimated based on the paths to all other vertices in G and G' that emanate from v_i and v'_j, correspondingly. First we order all end nodes in G following the clockwise contour with the starting node being v_i which we denote as v_{i0}. (Here we benefit from the fact that all skeleton endpoints lie on the contour.) We obtain a sequence of ordered end nodes $v_{i0}, v_{i1}, \ldots, v_{iK}$ in G, and similarly $v_{j0}, v_{j1}, \ldots, v_{jN}$ in G'. Then we compute the path distances between the two sequences (They represent the paths emanating from $v_i = v_{i0}$ in G and $v'_j = v'_{j0}$ in G'). We obtain a matrix of the path distances computed with formula (3):

$pd(v_i, v'_j) =$

$$\begin{pmatrix} pd(p(v_{i0},v_{i1}),p(v'_{j0},v'_{j1})) & pd(p(v_{i0},v_{i1}),p(v'_{j0},v'_{j2})) & \ldots & pd(p(v_{i0},v_{i1}),p(v'_{j0},v'_{jN})) \\ pd(p(v_{i0},v_{i2}),p(v'_{j0},v'_{j1})) & pd(p(v_{i0},v_{i2}),p(v'_{j0},v'_{j2})) & \ldots & pd(p(v_{i0},v_{i2}),p(v'_{j0},v'_{jN})) \\ \vdots & \vdots & & \vdots \\ pd(p(v_{i0},v_{iK}),p(v'_{j0},v'_{j1})) & pd(p(v_{i0},v_{iK}),p(v'_{j0},v'_{j2})) & \ldots & pd(p(v_{i0},v_{iK}),p(v'_{j0},v'_{jN})) \end{pmatrix} \quad (4)$$

To compute the dissimilarity value between the two end nodes v_i and v'_j, we extended the partial similarity of sequences method introduced in [36]. The extended method, which is described in Section 5, is called optimal subsequence bijection (OSB). The main property of OSB is the fact that it can skip outlier elements of matched sequences, which in our case means skipping some of the skeleton endpoints. For example, endpoint 7 in Fig. 5 must be skipped in order to establish the correct correspondence of the other skeleton endpoints. By applying OSB to the matrix in (4), we obtain the dissimilarity of two end nodes v_i and v'_j:

$$c(v_i, v'_j) = OSB(pd(v_i, v'_j)). \quad (5)$$

For two graphs G and G', with end nodes $v_i (i = 0, 1, 2, \ldots, K)$ and $v'_j (j = 0, 1, 2, \ldots, N)$, we compute all the dissimilarity costs between their end nodes and obtain a new matrix:

$$C(G, G') = \begin{pmatrix} c(v_0, v'_0) & c(v_0, v'_1) & \ldots & c(v_0, v'_N) \\ c(v_1, v'_0) & c(v_1, v'_1) & \ldots & c(v_1, v'_N) \\ \vdots & \vdots & \vdots & \vdots \\ c(v_K, v'_0) & c(v_K, v'_1) & \ldots & c(v_K, v'_N) \end{pmatrix} \quad (6)$$

Finally, we compute the total dissimilarity $c(G, G')$ between G and with the Hungarian algorithm on $C(G, G')$. For each end node v_i in G, the Hungarian algorithm can find its corresponding end node v'_j in G'. Since G and G' may have different numbers of end nodes, the total dissimilarity value should include the penalty for end nodes that did not find any partner. To achieve this, we simply add additional rows with a constant value const to (6) so that $C(G, G')$ becomes a square matrix. The constant value const is the average of all the other values in $C(G, G')$. The intuition for using the Hungarian algorithm is that we want to have a globally consistent one-to-one assignment of all end nodes with possibly assigning some end nodes to const, which represents a dummy node. This means that we seek a one-to-one correspondence of the end nodes in the skeleton graphs (with possibly skipping some nodes by assigning them to a dummy node).

Observe that our approach does not require any correspondence of junction nodes. This is extremely important, since as illustrated in Fig. 1, in many cases the correspondence of junction nodes is impossible to establish directly, and therefore, graph or tree editing approaches are needed if the correspondence of junction nodes is required. It is also important to observe that it is impossible to change the structure of junction nodes with skeleton pruning without eliminating some important end nodes. On the other hand, skeleton pruning is able to reduce the set of end nodes to structurally relevant nodes by eliminating spurious end nodes, see [34]. To summarize, the proposed skeleton graph matching is based on the assumption that similar skeletons have a similar structure of their end nodes that is measured by the similarity of shortest paths to other end nodes. This assumption is significantly weaker than the standard assumption that a structure of both end nodes and junction nodes is similar. Usually, the structure of both end nodes and junction nodes is weighted and edited, since as pointed out above (Fig. 1) it is common that skeletons of similar shapes have a different structure of junction nodes.

The fact that the Hungarian algorithm does not preserve the order of matched sequences does not influence the final score, since we can change the order only for similar end nodes. However, the similarity of end nodes is computed in the context of all other end nodes. Therefore, changing the order is likely due only to symmetry, in which case the final dissimilarity score is unaffected. The Hungarian algorithm has a computational advantage in comparison to order preserving assignment algorithms. When using an order preserving algorithm, we would need to enumerate over different starting nodes, while this is not necessary for the Hungarian algorithm. The Hungarian algorithm is the most popular for finding a maximum matching in a bipartite graph and is a common formulation for globally optimal matching. Examples include Kim and Kak [49], Siddiqi et al. [18] and, more recently, Belongie et al. [5], to name just a few; these techniques are also unable to enforce global ordering and are confused by object symmetries. A different way to approach the matching problem by allowing many-to-many mapping as proposed in [27] is also possible.

We give now a simple example illustrating our matching approach. Fig. 5 shows skeletons of two different horses with the corresponding end nodes linked by lines. We indexed the nodes so that the corresponding nodes have the same index except for node 7 that does not have a partner, and consequently corresponds to a dummy node with the correspondence value of const. The matrix $C(G, G')$ is shown in Table 1. The matching costs between most similar end nodes are marked with red numbers. The last row represents the dummy node.

V. Optimal subsequence bijection

The new algorithm, called Optimal Subsequence Bijection (OSB), works for elastic matching of two sequences of different lengths m and n. More specifically, for two finite sequences of end nodes of skeletons $a = (a_1, \ldots, a_m)$ and $b = (b_1, \ldots, b_n)$ in this paper. The goal is to find subsequences a' of a and b' of b such that a' best matches b'. Skipping (not matching) some elements of a and b is necessary, because both sequences may contain some outlier elements. However, skipping too many elements of sequence a increases a chance of accidental matches. To prevent this from happening, we introduce a penalty for skipping elements. The penalty can be expressed as matching

TABLE I

THE MATRIX $C(G, G')$ OF THE DISSIMILARITY VALUES BETWEEN ALL END NODES OF THE TWO HORSES IN FIG. 5. THE LAST ROW IS ADDED TO MAKE $C(G, G')$ A SQUARE MATRIX. THE VALUES IN RED ARE BETWEEN THE MOST SIMILAR END NODES.

	1	2	3	4	5	6	7
1	0.912	4.331	8.805	4.317	7.132	6.165	8.841
2	3.926	0.628	2.740	3.603	2.413	5.870	13.36
3	6.110	1.027	0.512	4.285	1.994	4.295	11.77
4	4.735	4.050	5.783	1.264	3.067	6.592	15.56
5	6.334	2.810	3.407	3.093	0.952	4.040	10.53
6	4.308	4.242	3.908	4.514	3.862	0.605	5.656
7	const	const	const	const	const	const	const

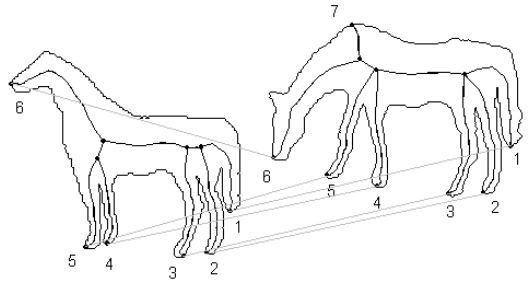


Fig. 5. The corresponding end nodes between the two skeleton graphs are linked with lines.

to some additional element b_∞ . Hence we extend sequence b by one more element b_∞ .

Our goal is to find the best possible correspondence of sequence a to a subsequence b' of b. We define a ***correspondence*** $f : \{1, \ldots, m\} \to \{1, \ldots, n, \infty\}$ as a monotonic injection for the restricted range of function $f : \{1, \ldots, m\} \to \{1, \ldots, n\}$ i.e., a function f such that $f(i) < f(i+1) < \infty$, where a_i is mapped to $b_{f(i)}$ for all $i \in \{1, \ldots, m\}$, and we allow a many-to-one mapping to ∞ . The assignment $f(i) = \infty$ means that we skip the element i in the sequence a. The sets of indices (i_k) and $(f(i_k))$ such that $f(i_k) < \infty$ for $i_k \in \{1, \ldots, m\}$ define the subsequences a' of a and b' of b, such that f restricted to (i_k) is a bijection. This explains the phrase "subsequence bijection" in **Optimal Subsequence Bijection** (OSB). However, we still need to define what optimal means here. We assume that the distance function d is given that can compute the dissimilarity value between any elements of a and b, i.e., $d(a_i, b_j)$ is given for $(i, j) \in \{1, \ldots, m\} \times \{1, \ldots, n, \infty\}$. We do not have any restrictions on the distance function d, and therefore any distance function is possible. In this paper we use the path distance pd defined in formula (3) as the distance d. While in most applications $d(a_i, b_j)$ is given for $(i, j) \in \{1, \ldots, m\} \times \{1, \ldots, n\}$, the distances to the additional element $d(a_i, b_\infty)$ should be carefully selected. Usually $d(a_i, b_\infty)$ is a constant for all $i \in \{1, \ldots, m\}$ that determines the cost of skipping any given element in sequence a. We call this constant $jumpcost$. In this paper, $d(a_i, b_\infty) = jumpcost$ is computed as

$$jumpcost = mean_i(min_j(d(a_i, b_j)) + std_i(min_j(d(a_i, b_j)). \quad (7)$$

Thus, every element a_i finds the closest element b_j, and then we take mean plus one standard deviation (std) of the distances to the closest elements. For example, if sequences a and b are similar with the exception of one outlier element, call it a_k, then every a_i for $i \neq k$ finds an element b_j with a small distance $d(a_i, b_j)$. Consequently the $jumpcost$ will be small, so that the distance to the closest element in b for a_k will be greater than the $jumpcost$, and the element a_k will be excluded from the matching with a relatively small penalty, i.e., we jump over it. For any given correspondence, we can define the distance between two sequences as:

$$d(a, b, f) = \frac{1}{m} \sum_{i=1}^{m} d(a_i, b_{f(i)})^2. \quad (8)$$

Our goal is to find a correspondence f so that $d(a', b', f)$ is minimal. More precisely, an optimal correspondence $\widehat{f}$ of elements in sequence a to elements in sequence b is defined as the one that yields the global minimum of $d(a, b, f)$ over all possible correspondences f:

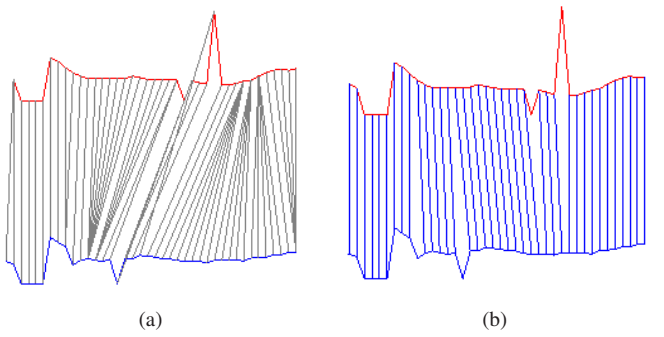
$$\widehat{f} = arg\ min\{d(a, b, f) : f\ is\ a\ correspondence\}. \quad (9)$$

Finally, the optimal distance is given by formula (8) for $f = \widehat{f}$. The optimal correspondence can be found with a shortest path algorithm on a DAG (directed acyclic graph). We denote the optimal distance $d(a, b)$ in (10) with $OSB(a, b)$ for the Optimal Subsequence Bijection distance. The nodes of the DAG are all index pairs $(i, j) \in \{1, \ldots, m\} \times \{1, \ldots, n\}$ and the edge costs w are defined as

$$w((i, j), (k, l)) = \begin{cases} d(a_i, b_j) & \text{if } i + 1 = k \text{ and } j + 1 \leq l \\ (k - i - 1) \cdot jumpcost & \text{if } i + 1 < k \text{ and } j + 1 \leq l \\ \infty & \text{otherwise} \end{cases} \quad (10)$$

Observe that there is no explicit penalty for skipping an element of sequence b. The intuition is that we expect all elements of sequence a to find a partner element in sequence b with possibly skipping some elements of b. An interesting situation occurs if both sequences a and b have an equal number of elements or sequence a is longer. Since each correspondence f is a injection on a, skipping an element of b implies skipping an element of a (with a penalty), and consequently, there is an implied penalty for skipping any element of b.

The OSB differs from the method in [36] in that it allows penalized skipping of outliers in the query sequence in addition



(a) (b)

Fig. 6. The links illustrate corresponding elements obtained with the proposed method OSB in (b) compared to the classical Dynamic Time Warping (DTW) in (a). Observe how outliers corrupt the result of DTW.

to skipping outliers in the target sequence. The method in [36] allows only skipping outlier elements in the target sequence.

To illustrate the benefit of the proposed one-to-one matching with skipping the outliers, consider the matching of the two real sequences shown in Fig. 6. Both the query sequence on top and the target sequence at the bottom have two outlier elements (shown as spikes). The proposed OSB method is able to skip them as illustrated in Fig. 6(b). For comparison, the correspondence obtained with Dynamic Time Warping (DTW) [39] shown in Fig. 6(a) is significantly corrupted by the outliers.

VI. Relation of OSB to other sequence similarity measures

The Dynamic Time Warping (DTW) distance has been shown to be superior to the Euclidean distance of sequences in many cases [40][41][42][43]. But, as illustrated in Fig. 6, DTW is particularly sensitive to outliers, since all elements of both sequences must participate in the correspondence.

The Longest Common Subsequence (LCSS) measure has been used in time series [44][45] to deal with the alignment and outliers problem. Given a query and a target series, LCSS determines their longest common subsequence, i.e., LCSS finds subsequences of the query and target (of the same length) that best correspond to each other. The distance is based on the ratio between the length of longest common subsequence and the length of the whole sequence. The subsequence does not need to consist of consecutive points, the order of points is not rearranged, and some points can remain unmatched. When LCSS is applied to sequences of numeric values, one needs to set a threshold that determines when values of corresponding points are treated as equal [45]. The performance of LCSS depends heavily on the correct setting of this threshold, which may be a particularly difficult problem for some applications.

The proposed OSB computes the distance value between two sequences based directly on the distances of corresponding elements, just as DTW does, and it allows the query sequence to match to only a subsequence of the target sequence, just as LCSS does. The main difference between LCSS and OSB is that LCSS optimizes over the length of the longest common subsequence (which requires a distance threshold), while OSB directly optimizes the sum of distances of corresponding elements. The main difference between DTW and OSB is that OSB can skip some elements of the target sequence when computing the correspondence while DTW requires that each point of the query sequence be matched to a point in the target sequence and vice versa. The main difference between OSB and the Hungarian algorithm is that the Hungarian algorithm does not preserve the order of sequences.

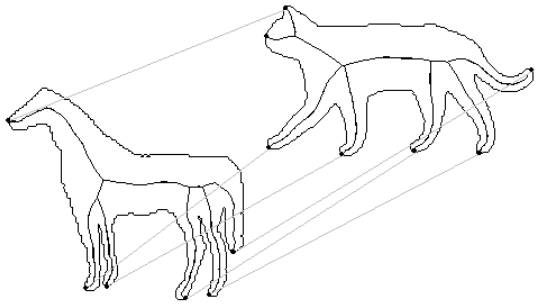


Fig. 7. The correspondence between a horse and a cat.

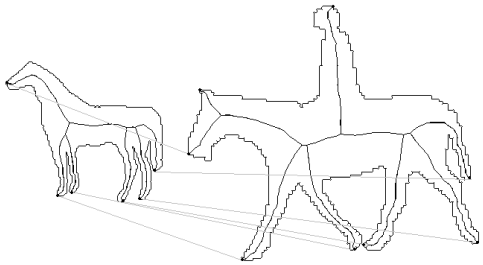


Fig. 8. The correspondence between a horse and a horse ridden by a person.

VII. Experiments

In this section, we evaluate the performance of the proposed method in two parts: (Section 7.1) matching the end nodes in the skeleton graphs, and (Section 7.2) the recognition performance of our method on standard shape databases. We illustrate in Section 7.3, the importance of matching skeleton graph structures as opposed to matching only tree structures.

A. *Correspondence matching*

Besides the matching of the two horses in Fig. 1, we tested the matching process on several other examples. In Fig. 7, we match a horse from Fig. 1 to a cat. Since the articulations of the horse are similar to the ones of the cat, our matching process finds the correct correspondences. Fig. 8 shows the correspondence between the end nodes of a horse in Fig. 5 and an outline of horse ridden by a person, which is similar to an example in [7]. Observe that the skeleton branch representing the person on the horse does affect the performance of the proposed approach. Fig. 8 demonstrates that the proposed method is able to establish a correct correspondence if parts are substantially altered. The two persons in the left part of Fig. 9 are taken from Kimia's dataset [14] while the silhouette on the right hand side is manually modified. Fig. 9 illustrates that the proposed approach works correctly if object parts are significantly altered (shortened in this case). We also show another matching result on hands from Kimia's dataset in Fig. 10. The obtained correspondence demonstrates that our matching process has strong performance in the presence of occlusion.

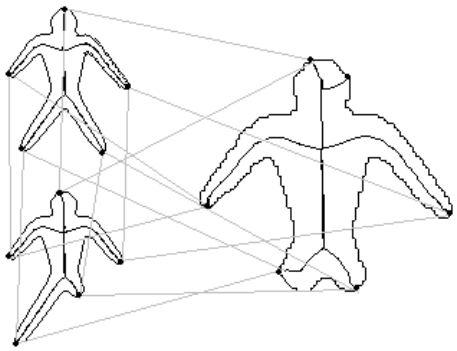


Fig. 9. The correspondence between the persons with different numbers of legs.

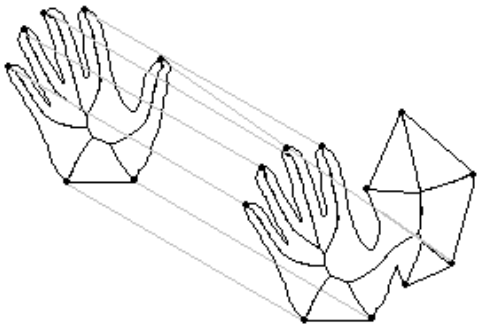


Fig. 10. The correspondence between a hand and another hand with occlusion.

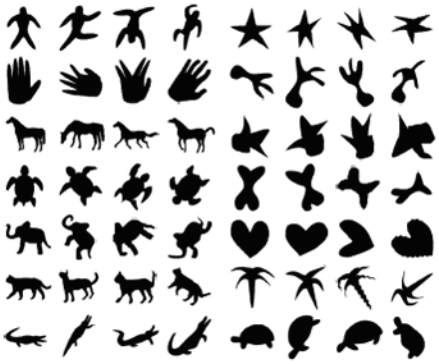


Fig. 11. Aslan and Tari database [21] with 56 shapes.

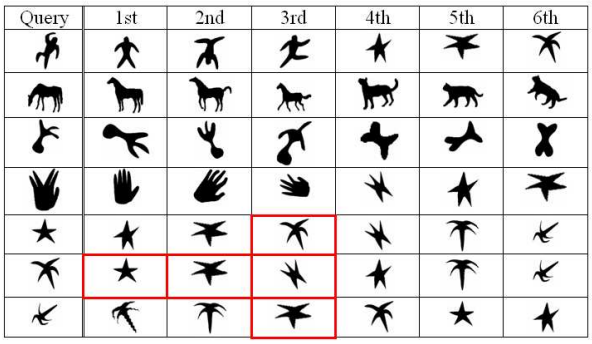


Fig. 12. Selected results of the proposed method on Aslan and Tari database [21]. Since each class is composed of 4 shapes, the query and the first 3 most similar shapes should be in the same class. Red boxes mark the results where this is not the case.

B. *Robustness of recognition*

To evaluate the recognition performance of our method, we tested it on four standard datasets: Aslan and Tari database [21], Kimia's two databases [15] and Rutgers tools database [16]. The Aslan and Tari database is used for testing the performance on non-rigid objects. As shown in Fig. 11, it includes 14 classes of articulated shapes with 4 shapes in each class. We use each shape in this database as a query. Several representative results are shown in Fig. 12, where six most similar shapes are shown for the queries. For each query, a perfect result should have the three most similar shapes in the same class as the query. The red squares mark all the results where this was not the case. Since there are only 5 errors in 168 query results, the recognition rate on this dataset (using the standard percent measure) is 97.0%. Using the bulls-eye test [32], the recognition rate is 99.4%. Moreover, we can easily observe that the wrong results are very similar to the query. For this dataset, we use parameters $M = 50$ and $\alpha = 40$. Although this database was introduced in Aslan and Tari [21], their paper does not present results on the whole database, as we do. ([21] illustrates only a few example dissimilarities between pairs of shapes). Since no recognition rate on the whole dataset is provided in [21], we cannot directly compare the recognition rate of our method to [21]. We were able to compare our method to the inner distance [47] on this dataset. We compared it to the best performing version in [47], which is the inner distance shape context with dynamic programming, denoted by IDSC + DP. The retrieval results are summarized as the number of correct shapes for all 56 queries among the 1st, 2nd and 3rd closest matches. IDSC + DP obtained 53, 51, 38 while our method achieved 55, 55, 53. The perfect result would be 56, 56, 56. We found that inner distance has problems with non-rigid deformations like bending of an arm. In contrast, the proposed method is designed to perform well in the presence of non-rigid deformations.

We also tested our algorithm on two shape databases provided by Kimia [15]. The first database as shown in Fig. 13 contains 216 images from 18 classes, which is a subset of MPEG-7 database [32]. For each shape, we check whether the 11 closest matches are in the same class as the query. In Table 2, we compare our result to two typical shape classification methods, and the number of correct matches in each rank is summarized. Our algorithm performs better than Shock-Edit [16] and Shape Context [5]. We use parameters $M = 50$ and $\alpha = 70$.

Fig. 14 shows another Kimia's dataset with 99 images from 9 classes. In this dataset, some images have protrusions or missing parts. Table 3 compares our results to several other methods in

Fig. 13. Sample shapes from Kimia's 216 shape database [15]. We show two shapes for each of the 18 classes.

TABLE II
RETRIEVAL RESULTS ON KIMIA'S 216 SHAPE DATABASE.

Algorithm	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th	11th
SC [5]	214	209	205	197	191	178	161	144	131	101	78
Shock Edit [15]	216	216	216	215	210	210	207	204	200	187	163
Our method	216	216	215	216	213	210	210	207	205	191	177

Fig. 14. Sample shapes from Kimia's 216 shape database [15]. We show two shapes for each of the 18 classes.

Fig. 15. Sample shapes in Rutgers Tools Database.

a way similar to Table 2. Our results are acceptable but are not the best, which is due to the presence of protrusions. While small protrusions do not present any problems, significant ones do. We illustrate the limitation of our method on the example in Fig. 10, where both shapes are taken form this dataset. We are able to compute the correct correspondence but the dissimilarity value is relatively large due to the necessity of skipping five skeleton endpoints in the protrusion. We use parameters $M = 50$ and $\alpha = 30$.

To compare our method to other typical skeleton-based approaches, we use the Rutgers Tools Database [16], which consists of 25 shapes grouped into 8 classes. Several sample shapes from the Rutgers Tools Database are shown in Fig. 15. Here we use parameters $M = 50$ and $\alpha = 40$. The results of the proposed method on the Rutgers Tools Database are shown in Fig. 16. We only have two mismatched entries, which are highlighted with red squares in the class 'Pliers' and the class 'Screwdriver'. Compared with other skeleton based methods, our method outperforms Shock Tree [11] (5 mismatched entries), Graph-Edit Distance [15] (5 mismatched entries), and Many-to-Many Matching [27] (3 mismatched entries). It is important to observe that skeleton-based methods significantly outperform contour-based methods on this dataset. As reported in [15], one of the most popular contour-based methods, Shape Contexts [5], misclassified 21 entries on this dataset.

C. Matching skeleton graphs that are not trees

This section illustrates the advantage of matching directly skeleton graphs as opposed to matching skeleton trees. Many approaches presented in the literature (e.g., Shock Tree [11], Many-to-Many Matching [27]) are unable to match skeleton graphs. They require first converting skeleton graphs to trees. However, as we illustrate now, this may result in losing important structural information. Fig. 17(a) shows two binary masks of keys. Observe that the holes are a characteristic shape feature of most keys.

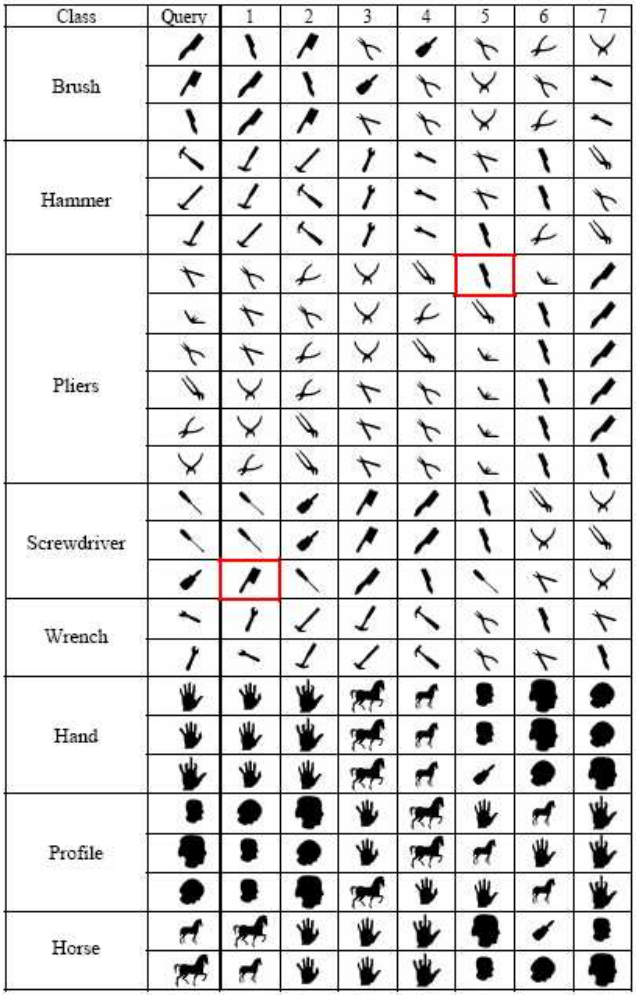


Fig. 16. The results of the proposed method on Rutgers Tools Database. We only have two mismatched entries which are highlighted with red squares.

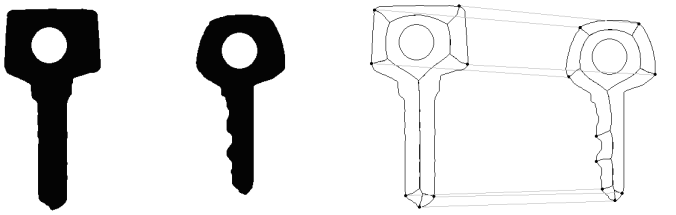
Fig. 17(b) shows their skeleton graphs and the correspondence between their end nodes computed by our algorithm.

For algorithms that are able to match only tree structures, it is necessary to convert the key graphs to trees by removing one of the edges in the loop. If, for example, the top edge in the loop is removed, the obtained tree structure becomes very similar to the skeleton of the wrenches shown in Fig. 18. Therefore, converting graphs to trees may result in the loss of important structural information, and consequently in the inability to correctly differentiate shapes.

To evaluate the performance of our algorithm on distinguishing the topological difference, we use a small database that contains five shapes: the two keys in Fig. 17, the two wrenches in Fig. 18 and the broken key in Fig. 19 (its skeleton is very similar

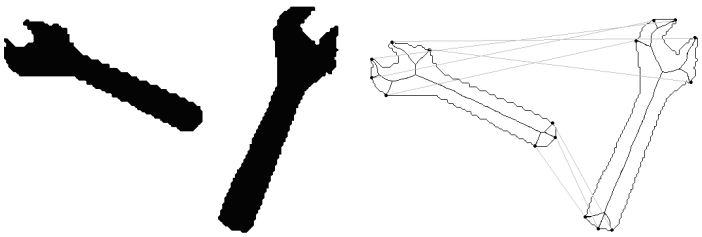
TABLE III
RETRIEVAL RESULTS ON KIMIA'S 99 SHAPE DATABASE.

Algorithm	1rd	2nd	3rd	4th	5th	6th	7th	8th	9th	10th
SC [5]	97	91	88	85	84	77	75	66	56	37
Gen. Model [48]	99	97	99	98	96	96	94	83	75	48
Our method	99	99	99	99	96	97	95	93	89	73
Shock Edit [15]	99	99	99	98	98	97	96	95	93	82
IDSC + DP [47]	99	99	99	98	98	97	97	98	94	79



(a) (b)

Fig. 17. The holes in the keys leads to skeletons that are graphs but not trees. The lines between end nodes illustrate the correspondence computed by our algorithm.



(a) (b)

Fig. 18. Two wrenches and their skeletons. The lines between end nodes illustrate the correspondence computed by our algorithm.

to the wrenches). The parameters for this database are $M = 50$ and $\alpha = 30$. Since the shortest paths between end nodes change dramatically when a loop is broken, we are able to distinguish the structural difference between a closed loop and a broken loop. Consequently, the broken key (without hole) is more similar to the wrenches than the keys (with holes) as shown in Fig. 20(a). It seems to be impossible for tree matching methods to capture this difference since they need to cut the loops on the skeletons before matching (in order to convert a graph to a tree structure). Therefore, we do not see how they could capture the topological difference between the broken key and the two unbroken keys. In analogy, we expect contour-based methods (e.g., [5][31][47][48]) to be unable to capture this difference too. To verify this claim we evaluated one of the best performing contour-based methods on this dataset. The results of IDSC + DP [47] on this database are shown in Fig. 20(b). In particular, the last row in Fig. 20(b) shows that IDSC + DP can not properly capture this topological difference.

It is important to mention that the proposed method requires the existence of end branches, which is always the case for polygonal shapes. However, ideal mathematical shapes like a doughnut may not have any end branches in which case the proposed method is not applicable.

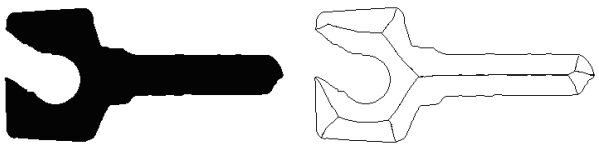


Fig. 19. A broken key and its skeleton.

VIII. IMPLEMENTATION AND COMPUTATIONAL COMPLEXITY

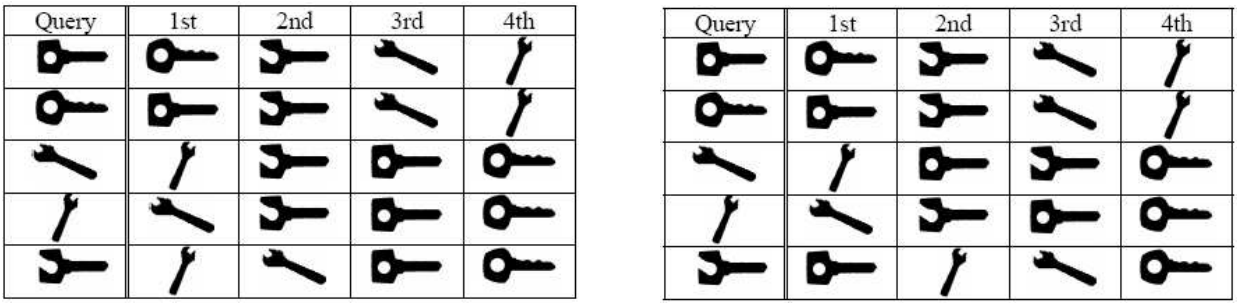
We briefly describe all the implementation steps: First, we compute skeletons with the algorithm in [28]. An important next step is proper skeleton pruning. We prune the skeletons with the algorithm in [34]. Then we find all the shape paths with Dijkstra's shortest path algorithm on the pruned skeleton graph. Finally, the total costs between skeleton graphs are computed with the proposed method.

We now analyze the computational complexity of the proposed graph matching approach. Let n_i be the number of end nodes in the skeleton graphs $G_i (i = 1, 2)$, and let m_i be the number of all nodes (including junction nodes) in G_i. Observer that $n_i < m_i$, since the number of end nodes is significantly smaller than the number of all nodes in the skeleton graph. Since in our experiment n_i and m_i are usually no more than 20, the time cost for comparing the similarity of two graphs is very small. The average time is approximately 0.015 seconds in our tests.

The time for Dijkstra's algorithm used to find the shortest path between two end modes on G_i is $O(m_i \log m_i)$, thus the time for computing all shape paths on G_i is $O(n_i^2 m_i \log m_i)$, since we have n_i^2 pairs of end nodes. The complexity for computing a shape distance between a pair of shape paths is $O(M)$, so the complexity for computing a path matrix is $O(n_1 n_2 M)$. Since the complexity of OSB is $O(n_1 n_2)$ [36], the time for computing the matrix $C(G_1, G_2)$ is $O(n_1^2 n_2^2)$. Since we assume $n_1 < n_2$, the time cost for the Hungarian algorithm in our paper is $O(n_2^3)$. Thus, the total complexity for our method is $O(n_i^2 m_i \log m_i) + O(n_1 n_2 M) + O(n_1^2 n_2^2) + O(n_2^3)$. Recalling that $n_i < m_i$, for i = 1, 2, by substituting the larger number m_i for all occurrences of n_i, our time complexity is bounded by $O(m_1^2 m_2^2)$. For example, this is two orders of magnitude smaller than the complexity of the Graph-Edit Distance, which is $O(m_1^3 m_2^3)$ [15].

IX. CONCLUSIONS

A novel object recognition method based on similarity of skeleton graphs is presented. The most significant contribution of



(a) (b)

Fig. 20. The results of the proposed method on a small database. Since we are able to distinguish the structural difference between a closed and open loop, the broken key (without hole) is more similar to the wrenches than the keys (with holes) in (a). Inner-Distance [47] can not capture this difference as shown in (b).

this paper is the novel approach to skeleton graph matching. We represent a skeleton as a set of geodesic paths between skeleton endpoints. The paths are compared using sequence matching. The proposed approach does not require any complicated strategies for tree/graph matching based on editing of topological structures and complicated weighting of branches/nodes. In addition to superior performance, the proposed method also reduces the time cost. Moreover, the fact that our representation of skeletons is based on their endpoints opens a possibility of new applications. We demonstrated one such application. We are able to compute the main symmetry axes of articulated objects by computing self similarity of skeleton divisions induced by pairs of endpoints.

The performance of our method is limited in the presence of large protrusions, since they require skipping a large number of skeleton endpoints. However, we believe this limitation can be solved with partial matching, e.g., when the dissimilarity is computed only for the pair of subgraphs that are most similar.

Our method is not limited to skeleton graphs. Our future work will also focus on matching any weighted graphs. In the case of planar graphs, we can still benefit from the cyclic order of end nodes. In the case of non-planar graphs, it appears to be possible to replace the cyclic order with the order of end nodes induced by the geodesic distance. We will also work on an efficient indexing scheme that is needed for fast, sub-linear database retrieval. Although our method is significantly faster than other skeleton graph matching approaches in direct comparison of two shapes, some of the existing methods allow for sub-linear database retrieval [15][25].

ACKNOWLEDGMENT

This work was support by the NSF under Grant No. IIS-0534929 and by the DOE under Grant No. DE-FG52-06NA27508. We wish to thank S. Dickinson and B.B. Kimia for kindly providing their databases. We also want to thank H. Ling for kindly providing his code for Inner-Distance method on the Internet.

REFERENCES

[1] H. Blum, "Biological Shape and Visual Science," *J. Theor. Biol.*, vol. 38: 205-287, 1973.

[2] T.B. Sebastian and B.B. Kimia, "Curves vs Skeletons in Object Recognition," *Signal Processing*, 85(2): 247-263, 2005.

[3] R. Basri, L. Costa, D. Geiger, and D. Jacobs, "Determining the Similarity of Deformable Shapes," *Vision Research*, 38: 2365-2385, 1998.

[4] D.P. Huttenlocher, G.A. Klanderman, and W.J. Rucklidge, "Comparing Images Using the Hausdorff Distance,"*IEEE Trans. Pattern Anal. Mach. Intell.*, 15(9): 850-863, 1993.

[5] S. Belongie, J. Puzhicha, and J. Malik, "Shape Matching and Object Recognition Using Shape Contexts,"*IEEE Trans. Pattern Anal. Mach. Intell.*, 24(4): 509-522, 2002.

[6] D. Shaked and A.M. Bruckstein, "Pruning Medial Axes,"*Computer Vision and Image Understanding*, 69(2): 156-169, 1998.

[7] S.C. Zhu and A.L. Yuille, "FORMS: A Flexible Object Recognition and Modeling System,"*Int. J. Comput. Vis.*, 20(3): 187-212, 1996.

[8] T. Liu, and D. Geiger, "Approximate Tree Matching and Shape Similarity," *Proc. Int. Conf. Computer Vision*, pp. 456-462, 1999.

[9] D. Geiger, T. Liu, and R.V. Kohn, "Representation and Self-similarity of Shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(1): 86-99, 2003.

[10] C. Di Ruberto, "Recognition of Shapes by Attributed Skeletal Graphs," *Pattern Recognition*, 37(1): 21-31, 2004.

[11] M. Pelillo, K. Siddiqi, and S.W. Zucker, "Matching Hierarchical Structures Using Association Graphs", *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(11): 1105-1120, 1999.

[12] M. Pelillo, "Matching Free Trees, Maximal Cliques, and Monotone Game Dynamics," *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(11): 1535-1541, 2002.

[13] B.B. Kimia, A.R. Tannenbaum, and S.W. Zucker, "Shape, Shocks and Deformations I: The Components of 2D Shape and the Reaction-Diffusion Space," *Int. J. Computer Vision*, 15(3): 189-224, 1995.

[14] T.B. Sebastian, P. Klein, and B.B. Kimia, "Recognition of Shapes by Editing Shock Graphs," In *ICCV*, pp. 755-762, 2001.

[15] T.B. Sebastian, P.N. Klein, and B.B. Kimia, "Recognition of Shapes by Editing their Shock Graphs," *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(5): 550-571, 2004.

[16] K. Siddiqi, A. Shkoufandeh, S. Dickinson and S. Zucker, "Shock Graphs and Shape Matching," In *ICCV*, pp. 222-229, 1998.

[17] K. Siddiqi, B. Kimia, A. Tannenbaum, and S. Zucker, "Shocks, Shapes, and Wiggles," *Image and Vision Computing*, 17(5-6): 365-373, 1999.

[18] K. Siddiqi, A. Shokoufandeh, S. Dickinson, and S. Zucker, "Shock Graphs and Shape Matching," *Int. J. Computer Vision*, 35(1): 13-32, 1999.

[19] K. Siddiqi and B.B. Kimia, "Parts of Visual Form: Computational Aspects," *IEEE Trans. Pattern Anal. Mach. Intell.*, 17(3): 239-251, 1995.

[20] M. Hilaga, Y. Shinagawa, T. Kohmura, and T.L. Kunii, "Topology Matching for Fully Automatic Similarity Estimation of 3D Shapes," In *ACM SIGGRAPH*, pp. 203-212, 2001.

[21] C. Aslan and S. Tari, "An Axis Based Representation for Recognition," In *ICCV*, pp. 1339-1346, 2005.

[22] A. Torsello and E.R. Hancock, "Computing Approximate Tree Edit Distance Using Relaxation Labeling," *Pattern Recognition Letters*, 24(8): 1089-1097, 2003.

[23] A. Torsello, E.R. Hancock,"A Skeletal Measure of 2D Shape Similarity," *Computer Vision and Image Understanding*, 95(1): 1-29, 2004.

[24] A. Torsello, D. Hidovic-Rowe, M. Pelillo, "Polynomial-Time Metrics for Attributed Trees". *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(7):1087-1099, 2005.

[25] A. Shokoufandeh, D. Macrini, S. Dickinson, K. Siddiqi, and S.W. Zucker, "Indexing Hierarchical Structures Using Graph Spectra," *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(7): 1125-1140, 2005.

[26] F. Demirci, A. Shokoufandeh, S. Dickinson, Y. Keselman, and L.

Bretzner, "Many-to-many Feature Matching Using Spherical Coding of Directed Graphs," In *ECCV*, pp. 322-335, 2004.
[27] F. Demirci, A. Shokoufandeh, Y. Keselman, L. Bretzner, and S. Dickinson, "Object Recognition as Many-to-Many Feature Matching," *Int. J. Comput. Vis.*, 69(2): 203-222, 2006.
[28] W.-P. Choi, K.-M. Lam, and W.-C. Siu, "Extraction of the Euclidean Skeleton Based on a Connectivity Criterion," *Pattern Recognition*, 36(3): 721 - 729, 2003.
[29] H.I. Choi, S.W. Choi, and H.P. Moon, "Mathematical Theory of Medial Axis Transform," *Pacific Journal of Mathematics*, 181(1): 57-88, 1997.
[30] L.J. Latecki and R. Lakämper, "Convexity Rule for Shape Decomposition Based on Discrete Contour Evolution," *Computer Vision and Image Understanding*, 73(3): 441-454, 1999.
[31] L.J. Latecki, R. Lakämper, "Shape Similarity Measure Based on Correspondence of Visual Parts," *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(10): 1185-1190, 2000.
[32] L.J. Latecki, R. Lakämper, and U. Eckhardt, "Shape Descriptors for Non-rigid Shapes with a Single Closed Contour," *Proc. CVPR*, pp. 424-429, 2000.
[33] X. Bai, L.J. Latecki, and W.-Y. Liu, "Skeleton Pruning by Contour Partitioning," *Int. Conf. on Discrete Geometry for Computer Imagery*, pp. 567-579, 2006.
[34] X. Bai, L.J. Latecki, and W.-Y. Liu, "Skeleton Pruning by Contour Partitioning with Discrete Curve Evolution," *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(3): 449-462, 2007.
[35] R.L. Ogniewicz and O. Kübler, "Hierarchic Voronoi Skeletons," *Pattern Recognition*, 28(3): 343-359, 1995.
[36] L. J. Latecki, V. Megalooikonomou, Q. Wang, R. Lakämper, C. A. Ratanamahatana, and E. Keogh, "Partial Elastic Matching of Time Series," *IEEE Int. Conf. on Data Mining*, pp. 701-704, 2005.
[37] T. Sebastian, P. Klein, and B. Kimia, "On Aligning Curves," *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(1):116-125, 2003.
[38] C.M. Cyr and B.B. Kimia, "3D Object Recognition Using Shape Similarity-Based Aspect Graph," In *ICCV*, pp. 254-261, 2001.
[39] H. Sakoe and S. Chiba, "Dynamic Programming Algorithm Optimization for Spoken Word Recognition," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 26(1): 43-49, 1978.
[40] J. Aach and G. Church, "Aligning Gene Expression Time Series with Time Warping Algorithms," *Bioinformatics*, 17(6): 495-508, 2001.
[41] G. Kollios, M. Vlachos, and D. Gunopoulos, "Discovering Similar Multidimensional Trajectories," *Proc. Int. Conf. on Data Engineering*, 673-684, 2002.
[42] S. Chu, E. Keogh, D. Hart, and M. Pazzani, "Iterative Deepening Dynamic Time Warping for Time Series," *Proc. SIAM Int. Conf. on Data Mining*, 2002.
[43] B. Yi, K. Jagadish, and C. Faloutsos, "Efficient Retrieval of Similar Time Sequences under Time Warping," *Proc. Int. Conf. on Data Engineering*, pp. 23-27, 1998.
[44] G. Das, D. Gunopoulos, and H. Mannila, "Finding Similar Timie Series," *Proc. PKDD Symposium*, pp. 88-100, 1997.
[45] M. Vlachos, M. Hadjieleftheriou, D. Gunopoulos, and E. Keogh, "Indexing Multi-dimensional Time-series with Support for Multiple Distance Measures," *Proc. of ACM SIGKDD*, 216-225, 2003.
[46] J. August, K. Siddiqi, and S. Zucker, "Ligature Instabilities and the Perceptual Organization of Shape," *Computer Vision and Image Understanding*, 76(3): 231-243, 1999.
[47] H. Ling and D.W. Jacobs, "Shape Classification Using Inner-Distance," *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(2): 286-299, 2007.
[48] Z. Tu and A. Yuille, "Shape Matching and Recognition: Using Generative Models and Informative Features," In *ECCV*, 3: 195-209, 2004.
[49] W.Y. Kim and A.C. Kak, "3-D Object Recognition Using Bipartite Matching Embedded in Discrete Relaxation," *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(3): 224-251, 1991.
[50] X. Bai, X. Yang, and L.J. Latecki, " Skeleton-based Shape Classification Using Path Similarity," *International Journal of Pattern Recognition and Artificial Intelligence*, to appear.

Xiang Bai received BS in electronics and information engineering from Huazhong University of Science & Technology (HUST), Wuhan, China, in 2003 and MS in electronics and information engineering from HUST in 2005. After Sept. 2005, he is a PhD candidate at the Department of Electronics and Information Engineering of HUST. From Jan. 2006 to May 2007, he worked in Department of Computer Science and Information of Temple University. Currently, he has joined the Department of Neurology of University of California, Los Angeles as a joint PhD student. His research interests include computer graphics, computer vision and pattern recognition.

Longin Jan Latecki is the winner of the Pattern Recognition Society Award together with Azriel Rosenfeld for the best article published in Pattern Recognition in 1998. He received the main annual award from the German Society for Pattern Recognition (DAGM), the 2000 Olympus Prize. He is an Editorial Board Member of Pattern Recognition. He co-chairs the IS&TSPIE annual conference series on Vision Geometry. He published over 150 research papers and books. He is an associate professor for computer science at Temple University in Philadelphia. He received a PhD in Computer Science from the Hamburg University in Germany in 1992. His main research areas are shape representation and similarity, robot perception, and digital geometry and topology.