**Appeared in the International Conference on Computer Vision, Kerkyra, Greece, 1999.**

# Approximate Tree Matching and Shape Similarity

Tyng-Luh Liu
Institute of Information Science
Academia Sinica
Nankang, Taipei 115 Taiwan
liutyng@iis.sinica.edu.tw

Davi Geiger
Courant Institute of Mathematical Science
New York University
New York, NY 10012 USA
geiger@cs.nyu.edu

## Abstract

*We present a framework for 2D shape contour (silhouette) comparison that can account for stretchings, occlusions and region information. Topological changes due to the original 3D scenarios and articulations are also addressed. To compare the degree of similarity between any two shapes, our approach is to represent each shape contour with a free tree structure derived from a shape axis (SA) model, which we have recently proposed. We then use a tree matching scheme to find the best approximate match and the matching cost. To deal with articulations, stretchings and occlusions, three local tree matching operations, merge, cut, and merge-and-cut, are introduced to yield optimally approximate matches, which can accommodate not only one-to-one but many-to-many mappings. The optimization process gives guaranteed globally optimal match efficiently. Experimental results on a variety of shape contours are provided.*

## 1 Introduction

Object shapes can deform due to changes in viewing condition, deformations, articulations and occlusions. In order to compare two object shapes, i.e., to assign a match/correspondence and to give a measure of similarity, all these issues must be modeled and accounted for.

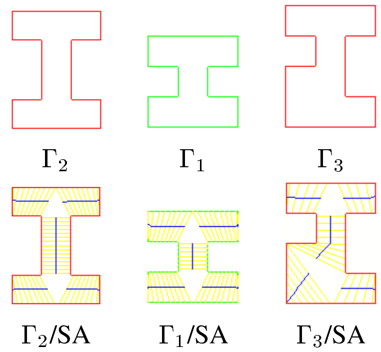
Methods to compare two shape contours based on evaluating global deformations [6] tend to be sensitive to occlusion and fail to account for local deformations (such as articulations).

A class of methods compares objects by deforming one object into another and evaluating the amount of deformation applied in this process, including [7, 19]. Guaranteed methods, typically, use dynamic programming (time-warping) to register two contours. These are all string (contour) matching algorithms, e.g., [2, 5]. Problems with these approaches are that they

1. do not account for region information and for symmetries (see Figure-1).

2. are sensitive to topological changes. Imagine comparing two flowers with a stem in different sides (see Figure-4(b)(d)). Most approaches will consider these as occlusions and need to pay for large penalties/costs to match them, while the fact that the "occluded" parts are similar though at different places.

3. have problems of efficiency, since the size of occlusions can make these methods drastically slow.




| COST | $\Gamma_1$ | $\Gamma_2$ | $\Gamma_3$ |
|---|---|---|---|
| $\Gamma_1$ | 0 | 11.2607 | 18.874 |
| $\Gamma_2$ | 11.2607 | 0 | 17.8634 |
| $\Gamma_3$ | 18.874 | 17.8634 | 0 |

**Figure 1.** **$\Gamma_2$ and $\Gamma_3$ are derived from $\Gamma_1$ by different deformations (same amount of stretchings but at different places). While most of the local string deformation methods will fail to distinguish the dissimilarity between them, $\Gamma_2$ is considered more similar to $\Gamma_1$ by our method.**

Our goal is to develop a shape representation of objects that yields similarity measures that can account for local deformations, symmetries, and region information. We follow the view of comparing deformable objects by measuring the amount of energy needed to locally deform one shape into the other. Moreover, we attempt to provide a shape comparison method that considers not only local deformations but global shape symmetries.

We start with the representation of shapes and consider a *shape axis* (SA) representation [9](see Figure-2). The SA of a given shape contour is obtained through a self-similarity variational framework; a unique *shape axis tree* (SA-tree), where every pair of consecutive nodes (edge) corresponds to an object substructure, can be constructed to encode the contour data and its SA. Since the SA framework is variational, we also obtain a measure of how effective the SA-tree representation is for a given shape, namely the value of the minimal cost.

Each shape contour is represented by an SA-tree so that the similarity between shapes can be evaluated via a tree matching scheme. The cost of matching edges is the cost of comparing two object parts (local deformations and region information can be considered). A key issue is to structure the set of possible correspondences, and perform an efficient search for the best correspondence. We also require a cost function to determine how local differences between the shape contours should affect their perceived similarity.

In this paper we describe a tree matching scheme that uses the neighboring topological structure among nodes and is much more complicated than simple string matching algorithms. Any two SA-trees are not required to have the same number of nodes; thus we seek the best approximate tree match between the two trees [14]. Pruning and merging vertices can be applied in the process of matching. Such a method has to address occlusions, that is, some subtrees of an SA-tree may not be matched. It has to account for region information and stretching, i.e., the comparison/matching can not only be between tree structures, but has to consider the region and contour segments associated with a particular edge of an SA-tree. It has to deal with articulations, e.g., the cost for the mismatch of angles (each angle is measured by a pair of consecutive edges) should increase sub-linearly. As we will show, the tree-matching shape comparison algorithm is very efficient and it can be applied to, e.g., animation and on-line image (shape contour) database retrieval.

### 1.1 Previous Work

Siddiqi *et al.* [17] have proposed a shape matching method based on a shock graph grammar where to match two nodes in the shock trees, an affine transformation is used to align two interpolated geometric curves. The approach is interesting but can not account for articulations occurred with respect to each node's geometric structure. More recently, they have presented a new framework based on finding maximal cliques of the association graphs to match two trees [12]. The matching scheme works well in matching hierarchical structures. However, it is limited to finding only one-to-one correspondences, which may not be suitable for flexible objects with articulations where an object part may correspond to more than one nodes.

In Zhu and Yuille's work [20], a FORMS system is proposed to recognize and represent flexible objects from their silhouettes. The silhouettes are derived from skeleton extraction and part segmentation, using a deformable circle method. To compare two objects, say hands, they first compute each object's skeleton then match each skeleton to a model of hand where its skeleton is well-defined. In this way, the skeleton of each object can be refined. A pair of parts in the two objects are matched to each other if they correspond to the same part in the model. This implies that the shape comparison between two object is not done directly but via referencing an additional model.

Our method in shape similarity differs from theirs on allowing many-to-many correspondences so an edge in an SA-tree can be matched to to a path consisting of more than one edges in the other SA-tree (note that, for simplicity, we only consider paths consisting of two consecutive edges). Thus the mappings between nodes of two SA-trees are not required to be one-to-one due to the merge, cut and merger-cut operations. Also, to compute the cost of an edge-to-edge or edge-to-path matching, we have used a local shape comparison model [2] that can account for articulations. This is important that we can easily extend our approach to segmentation for real images by combining this model with an active contour tracker. Unlike the FORMS, in our system no model is required when comparing two shapes to derive the correspondences.

## 2 Shape Representation

We adopt the shape representation framework developed in [9], leading to a unique SA-tree. The advantages over other related representations [3, 10, 1, 4, 11, 13, 15, 16] are that (i) we are not seeking a symmetry axis representation but rather, a set of correspondences along the shape contour structured in a tree graph, and (ii) we use a variational approach to establish a measurement on how good the representation of a contour shape is (see Figure-1).

### 2.1 Shape Axis Tree

Given a (shape) contour (e.g., Figure-2 (a)(b)(c)), we can represent it as a parameterized curve: $\Gamma = \Gamma(s) = \{x(s), s \in [0, 1)\}$ where $x(s)$ are the coordinates of the contour points. To find the SA of contour $\Gamma(s)$, we match

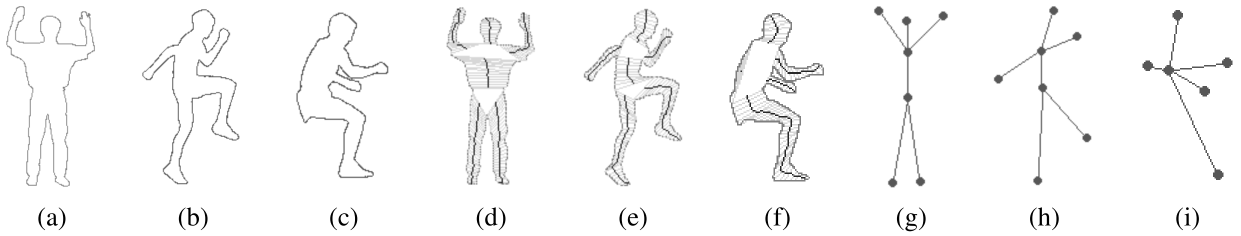



**Figure 2. Shape axis model: shape contours and their shape axes and SA-trees. In an SA-tree, the non-leaf vertices correspond to bifurcations in the shape axes.**

$\Gamma(s)$ to its own mirror version $\tilde{\Gamma} = \tilde{\Gamma}(t) = \{\tilde{x}(t) = x(1-t), t \in (0,1]\}$ using the cost functional established in [9]. Given a correspondence $t(s)$ between $\Gamma(s)$ and $\tilde{\Gamma}(t)$, the SA of contour $\Gamma$ is defined as the set of middle points between $x(s)$ and $\tilde{x}(t)$, i.e.,

$$x^{SA}(s) = \frac{x(s) + \tilde{x}(t(s))}{2} = \frac{x(s) + x(1 - t(s))}{2}.$$

Also following [9], we can construct a unique SA-tree by grouping the discontinuities in the the correspondence $t(s)$. In Figure-2(d)(e)(f), the dashed lines are the optimal correspondences $t(s)$ and the shape axes are formed by connecting the middle points. The corresponding SA-trees are shown in Figure-2 (g)(h)(i).

An SA-tree is a free tree (a connected, acyclic and undirected graph) and there are two types of vertices in $T$. The first type of vertices contains all the leaves and the second includes those corresponding to bifurcations (non-leaf vertices). Note that each edge of an SA-tree corresponds to a pair of shape contour segments (see Figure-2).

## 3 Shape Similarity and Tree Matching

Our approach makes use of both the global and local information of shapes. The global symmetries are captured by the SA-tree representation itself. The degree of deformation of one shape into the other is then modeled by the cost of approximately matching one SA-tree to the other one. An approximate matching is necessary since viewing position, occlusion and stretching may yield different SA-trees for the same object shape.

By formulating the shape similarity problem as an approximate tree matching one, we need to investigate the following issues.

- Unlike the regular approximate tree pattern matching [14], we want to find not only the node-to-node but also the edge-to-edge/edge-to-path correspondences. In our case, a node of an SA-tree conveys the topological structure of shape and an edge encodes the corresponding shape information.
- When modeling occlusions, we need to consider the possible deletions or merges of subtree structures and to estimate the penalties (or costs) for them.
- When modeling articulations and stretchings, we should have a local shape comparison model to evaluate the cost of edge-to-edge/edge-to-path matching so that it can account for articulations. By an edge-to-path matching, we mean a stretching matching over the tree structures. Note that while the comparison is fully structured by the SA-trees, the cost of comparing edges is based on the actual shapes associated with the edges. This cost includes shape bending and stretching and possibly other region deformations.

The optimal approximate matching between two SA-trees can be found efficiently with an $A^*$ algorithm. Our focus is to illustrate that with only local tree matching operations (to be defined later), the shape comparison method can account for topological changes, articulations, deformations and occlusions.

We now first elaborate how to find the best approximate matching then concentrate on how to formulate the local tree matching operations.

### 3.1 SA-Tree Matching Algorithm

Conceptually, we can construct a solution tree where each node represents an edge-to-edge or edge-to-path local matching of two SA-trees, and every path, from the root to a leaf, corresponds to a sequence of local matchings and a possible solution/match. Our goal is to find the best match, and this can be achieved by using an $A^*$-like algorithm to locate the optimal path in the solution tree, i.e., the best approximate match. Though there are many different ways to solve the approximate tree matching problems in polynomial time [14], the $A^*$ approach has the advantage to be easily extended to real image application.

To initialize the optimization process, we begin with setting up a "virtual root" of a solution tree with cost $0$. Then, generate all the root's children, i.e., the level-1 nodes, by considering all possible edge-to-edge or edge-to-path local matchings where each of them must contain a leaf. Add all the level-1 nodes into a priority queue $Q$ together with their respective (local) matching costs. To grow the solution tree, the current item in $Q$ with the minimal key (cost) is located and this min-item corresponds to some node in the solution tree. We then extend all of its possible child nodes and again add them into $Q$, taking into account the existing local matches from the root to this current min-node as well as properties of local matching. The optimization process stops when we first reach a leaf in the solution tree and the optimal path can be recovered by tracing back to the root.

Note that although an edge-to-edge or edge-to-path local matching may appear in many different paths in a solution tree, its shape comparison cost is only computed once, that is, we save the local shape comparison costs in a look-up table. This guarantees that our method can efficiently locate the best approximate matching.

### 3.2 Tree Matching Operations

We now explain how shape information is encoded into an SA-tree structure and how local tree matching operations are applied to deal with occlusions and stretchings. To illustrate, we use shape contours and SA-trees in Figure-3.

We denote the edge connecting vertices $u_2$ and $u_5$ as $e(u_2, u_5)$ and the corresponding contour segments as $CT(u_2, u_5)$ (see Figure-3 (b)) where, in this example, $CT(u_2, u_5) = [\Gamma_1^{BC}, \Gamma_1^{CD}]$. Note that the order of contour segments does matter and is always arranged in a counter-clockwise manner. This implies $CT(u_2, u_5) = CT(u_5, u_2) = [\Gamma_1^{BC}, \Gamma_1^{CD}]$.

Next we describe some useful rules/models that we have adopted for matching two contours via tree structures.

1. To compare the similarity between two contour segments, we use the model established in [2] to compute the cost, i.e., given two contour segments, say $\Gamma_s$ (parameterized by $s$) and $\Gamma_t$ (parameterized by $t$), the cost of shape similarity comparison is

$$cost_S(\Gamma_s, \Gamma_t) = \min_{t(s)} cost_S(\Gamma_s, \Gamma_t, t(s))$$
$$= \min_{t(s)} \int_{\Gamma_s} \left[ \frac{|k_t\, t' - k_s|^2}{|k_t\, t'| + |k_s|} + \lambda \frac{|t'-1|^2}{t'+1} \right] ds\,, \quad (1)$$

where $t' = dt/ds$ and $k_s,\ k_t$ are the curvatures at $\Gamma_s(s)$, $\Gamma_t(t)$, respectively. The first term in the integral of (1) is the bending cost and the second is the stretching cost. $\lambda$ weights the relative contributions of stretching and bending. A correspondence $t(s)$ is considered optimal if it minimizes (1). It is known that the above model can account for articulations.

2. Given two SA-trees, say $T_1$ and $T_2$, we say that $e(u_i, u_j) \in T_1$ is matched to $e(v_k, v_l) \in T_2$ if node $u_i$ is mapped to node $v_k$ and node $u_j$ is mapped to node $v_l$. The cost is denoted as $cost(e(u_i, u_j), e(v_k, v_l))$ and is computed from the cost of comparing $CT(u_i, u_j)$ with $CT(v_k, v_l)$, i.e.,

$$cost(e(u_i, u_j), e(v_k, v_l))$$
$$= cost_S(CT(u_i, u_j), CT(v_k, v_l))\,.$$

For example, in Figure-3 (a)(b)(c)(d), the cost of matching $e(u_5, u_2) \in T_1$ to $e(v_4, v_1) \in T_2$ is

$$cost_S(CT(u_5, u_2), CT(v_4, v_1))$$
$$= cost_S(\Gamma_1^{BC}, \Gamma_2^{AB}) + cost_S(\Gamma_1^{CD}, \Gamma_2^{BC})\,.$$

3. We require a leaf in $T_1$ can only be matched to a leaf in $T_2$, and vice versa. This gives rise to the topological similarity (this condition can be relaxed by allowing "stretching" matchings described next).

4. Recall that an SA-tree is a free tree. When comparing two SA-trees, they become rooted trees with respect to each solution path. For instance, in Figure-3, $u_5$ and $v_4$ become the root of $T_1$ and $T_2$, respectively, along a solution path starting with an initial local match between $e(u_5, u_2)$ and $e(v_4, v_1)$.

The above rules are not sufficient for our application. We need to incorporate tree matching operations that can cut or merge the substructures of a tree to derive a good match and account for deformations and occlusions. Thus we introduce local "stretching matchings" allowing an edge to be matched to a path of length 2. The notation $p(u, v, w)$ is used to denote a path of two edges, namely $e(u, v)$ followed by $e(v, w)$. There are three types of stretching matching introduced in this work including *merge*, *cut* and *merge-and-cut*. Altogether, they address the issues of stretchings and occlusions directly.

**Merge operation:** The structure of SA-trees from a same class of objects could be different due to movements and stretchings (see Figure-3 (b)(d)). To model the scenario, we design a "merge" operation ($M$-operation for abbreviation) that an edge, say $e(v_2, v_1)$ can be matched to, say, $p(u_3, u_2, u_1)$ through a merge between nodes $u_2$ and $u_1$ (Figure-3 (g)). The newly merged node will be denoted as $[u_2 u_1]$ to indicate that node $u_1$ is merged with $u_2$ and all child nodes of $u_1$ become children of $u_2$. For each merge, a penalty cost, denoted as $cost_M$, needs to be paid and it is

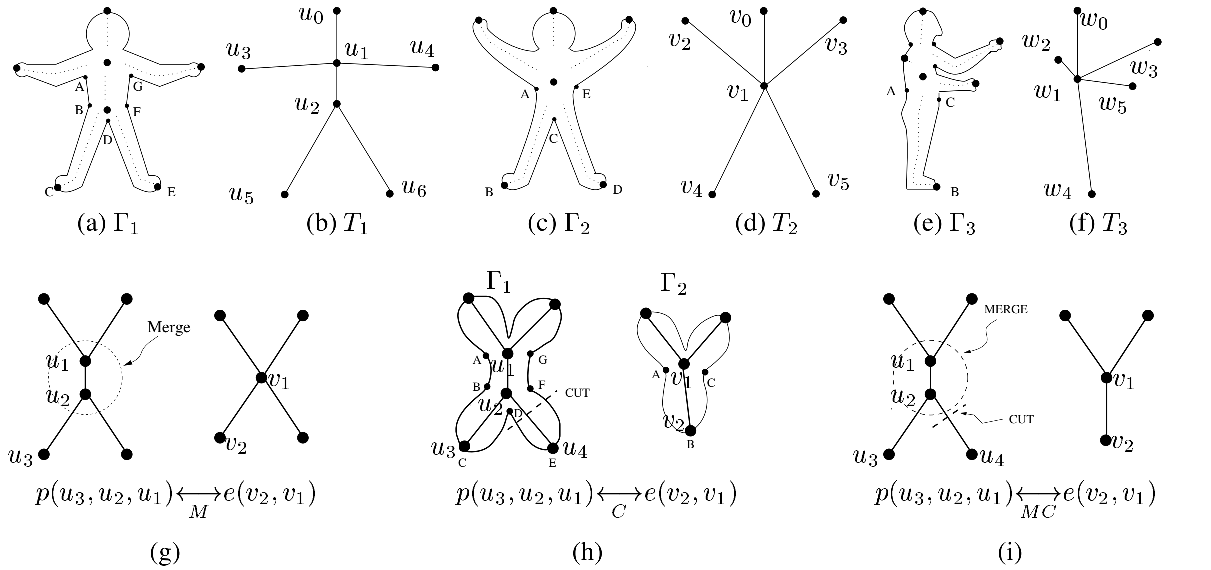



**Figure 3. (a)-(f) are human shape contours and their SA-trees. (g) An example of stretching match via a merge operation ($M$-operation). (h) An example of stretching match via a cut operation ($C$-operation) overlapped with its shape contour. (i) An example of stretching match via a merge-and-cut ($MC$-operation).**

proportional to the product of total length of contour segments being merged and some positive real-valued function of the difference of the neighboring configurations between the merged node, $[u_2 u_1]$, and its matched node, $v1$. More specifically, the total cost of this match via an $M$-operation is

$$\begin{aligned} &cost(p(u_3,[u_2u_1]),e(v_2,v_1)) \\ &= cost_S(CT(u_3,u_2),CT(v_2,v_1)) + cost_M([u_2u_1],v_1)\,, \end{aligned}$$

where

$$\begin{aligned} &cost_M([u_2u_1],v_1) \\ &= \alpha \times \Big| CT(u_2,u_1) \Big| \times (1 + \tfrac{|deg([u_2u_1]) - deg(v_1)|}{max(deg([u_2u_1]),deg(v_1))})\,. \end{aligned}$$

In most cases, the parameter $\alpha$ is set to 0.2. The notation $deg(u)$ is the number of adjacent nodes of $u$ and $\Big| CT(u_2,u_1) \Big|$ is the length of contour segments of $CT(u_2,u_1)$. In Figure-3 (g), we have $deg([u_2u_1]) = deg(v_1) = 4$. The penalty $cost_M$ is defined in the way that, after a merge, the less similar in the topological configurations are, the more expensive $cost_M$ is.

**Cut operation:** A "cut" operation ($C$-operation) is applied to a stretching match to remove extra subtree structures. This is especially useful in dealing with occlusions while some part structures of an object are missing due to changes of viewing direction.

In Figure-3 (h), $p(u_3,u_2,u_1)$ is matched to $e(v_2,v_1)$ via a "$C$-operation". We use the notation $u_3\hat{u}_2u_1$ to indicate that except $e(u_3,u_2)$ and $e(u_2,u_1)$, all edges (including their subtrees) connecting to $u_2$ are cut from the SA-tree. Similar to the merge case, we need to estimate the penalty cost, $cost_C$, for a $C$-operation.

$$\begin{aligned} &cost(p(u_3,\hat{u}_2,u_1),e(v_2,v_1)) \\ &= cost_S(CT(u_3,u_2) \cup CT(u_2,u_1), CT(v_2,v_1)) \\ &\quad + cost_C(u_3\hat{u}_2u_1)\,, \end{aligned}$$

where

$$\begin{aligned} &cost_S(CT(u_3,u_2) \cup CT(u_2,u_1), CT(v_2,v_1)) \\ &= cost_S(\Gamma_1^{AB} \cup \Gamma_1^{BC}, \Gamma_2^{AB}) + cost_S(\Gamma_1^{CD} \cup \Gamma_1^{FG}, \Gamma_2^{BC}) \end{aligned}$$

and

$$\begin{aligned} &cost_C(u_3\hat{u}_2u_1) \\ &= \beta \times (\Big| CT(u_2,u_4) \Big| + Gap(e(u_3,u_2),e(u_2,u_1)))\,. \end{aligned}$$

$Gap(e(u_3,u_2),e(u_2,u_1))$ is the length of total gaps between $CT(u_3,u_2)$ and $CT(u_2,u_1)$ and it is equal to the distance between point $D$ and $F$ in contour $\Gamma_1$ of Figure-3 (h). Again, $\beta$ is a parameter to be adjusted and we have used $\beta = 0.2$. Note that a $C$-operation is not just deleting some subtrees form a contour but also creating a gap (see Figure-4 (g) and Figure-5(e)). Thus, both factors should be considered in formulating a reasonable $cost_C$.

**Merge-and-Cut operation:** A merge-and-cut ($MC$-operation) is a combination of $M$-operation and $C$-operation as shown in Figure-3 (i). Therefore, the cost of a stretching match with an $MC$-operation is

$$cost(p(u_3, [\hat{u}_2 u_1)]), e(v_2, v_1))$$
$$= cost_S(CT(u_3, u_2), CT(v_2, v_1)) + cost_{MC}(u_3[\hat{u}_2 u_1]) \,,$$

where

$$cost_{MC}(u_3[\hat{u}_2 u_1]) = cost_M([u_2 u_1], v_1) + cost_C(u_3 \hat{u}_2 u_1) \,.$$

## 4 Examples and Discussion

Some of the experimental results are shown in Figure-4 and 5. In each example, we show the best approximate match between the two contours and the comparison cost. The quadruple includes the sizes of contours associated with the two SA-trees and the sizes of omitted contour segments of the best match due to cuts or merges. It takes less than one minute to complete a shape comparison task on a Pentium-II PC, if the SA structures are given.

We have developed a tree matching framework combining local and global approach for shape comparison based on a shape axis model. The issues of occlusions and articulations are handled by formulating the comparison task as an approximate tree matching problem. We use $A^*$ algorithm to find the comparison cost and best matching between a pair of contours. Our method can be extended to real images because (1) the region information can be used in modeling the tree matching operations, and (2) the $A^*$ scheme is easier to be combined with a tree structure-wise grouping process.

### Acknowledgments

T-L. Liu is supported in part by the Institute of Information Science, Academia Sinica of Taiwan. D. Geiger is supported in part by an NSF career award grant.

## References

[1] H. Asada and M. Brady. The Curvature Primal Sketch. *IEEE PAMI*, Vol. 5, pp. 2–14, 1983.

[2] B. Basri, L. Costa, D. Geiger, and D. Jacobs. Determine Shape Similarity. IEEE workshop in Physics Based Vision, Boston, June 1995.

[3] H. Blum. Biological Shape and Visual Science. *J. of Theoretical Biology*, 38:205-287, 1973.

[4] C.A. Burbeck and S.M. Pizer. Object Representation by Cores: Identifying and Representing Primitive Spatial Regions. *Vision Research*, Vol. 35, pp. 1917-19301995.

[5] Y. Gdalyahu and D. Weinshall. Measures for Silhouettes Resemblance and Representative Silhouettes of Curved Objects. *4th ECCV*, Cambridge, UK, April 1996.

[6] D. Huttenlocher, G. Klanderman, and W. Rucklidge. Comparing Images Using the Hausdorff Distance. *IEEE PAMI*, 15(9):850-863, 1993.

[7] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active Contour Models. *Int. J. Comput. Vision* 1(4):321-331, 1998.

[8] K. Kupeev and H. Wolfson. On Shape Similarity. *Proceedings Int. Conf. on Pattern Recognition*, pp. 227-237, 1994.

[9] T-L. Liu, D. Geiger and R. V. Kohn. Representation and Self-Similarity of Shapes. *ICCV*, pp. 1129-1135, Bombay, India, 1998.

[10] R. Nevatia and T. O. Binford. Description and Recognition of Curved Objects. *Artificial Intelligence*, Vol. 8, pp. 77–98, 1977.

[11] R. Ogniewicz. *Discrete Voronoi Skeletons*. Hartung-Gorre, 1993.

[12] M. Pelillo, K. Siddiqi and S. W. Zucker. Matching Hierarchical Structures Using Association Graphs. *ECCV*, Freiburg, Germany, 1998.

[13] W. Richards and D. D. Hoffman. Codon Constraints on Closed $2D$ Shapes. *CVGIP*, 31(2):156-177, 1985.

[14] D. Shasha, J. Wang and K. Zhang. Exact and Approximate Algorithm for Unordered Tree Matching. IEEE Trans. Systems, Man, and Cybernetics, 24(4), pp. 668-678, 1994.

[15] K. Siddiqi and B. B. Kimia. Parts of Visual Form: Computational Aspects. *IEEE PAMI*, Vol. 17, No. 3, pp. 239-251, March, 1995.

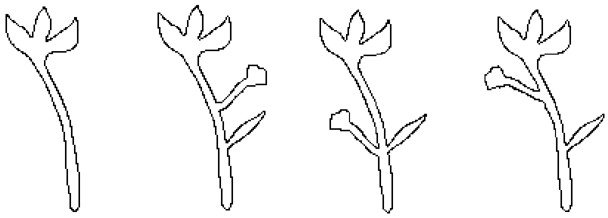
[16] K. Siddiqi and B.B. Kimia. A Shock Grammar for Recognition. *CVPR*, pp. 507-513, S. Francisco, 1996.

[17] K. Siddiqi, A. Shokoufandeh, S. Dickinson and S.Zucker. Shock Graphs and Shape Matching. *ICCV*, Bombay, India, 1998.

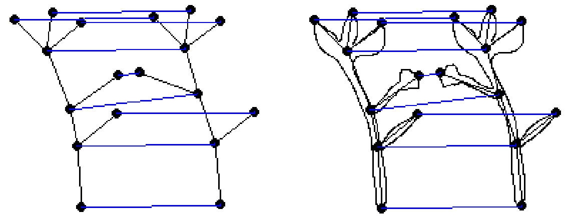
[18] D. Terzopolous, A. Witkin,A. and. M. Kass. Symmetry-seeking models and 3D object recovery. Int. J. Comput. Vision, 1, pp. 211-221, 1987.

[19] S. Ullman. Aligning Pictorial Descriptions: An Approach to Object Recognition. Cognition, 32(3):193-254, 1989.

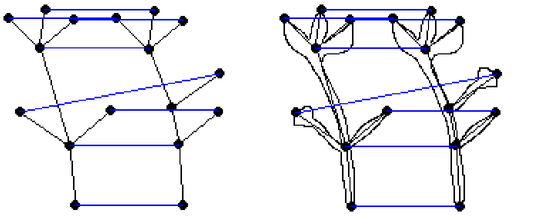
[20] S. C. Zhu and A. L. Yuille. FORMS: a Flexible Object Recognition and Modeling System. *ICCV*, Boston, 1995.



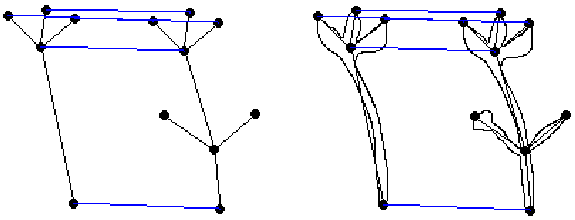
(a) $F_1$ (412) (b) $F_2$ (601) (c) $F_3$ (601) (d) $F_4$ (613)



(e) Cost: $25.2766$ $(601, 613, 0, 0)$ (exact matching)

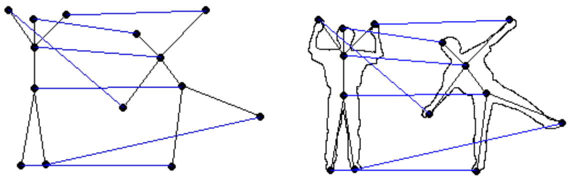


(f) Cost: $43.3587$ $(601, 601, 0, 52)$ (merge)

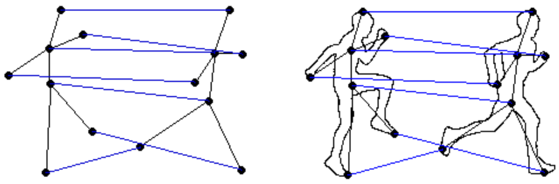


(g) Cost: $63.561$ $(422, 601, 0, 277)$ (cut)

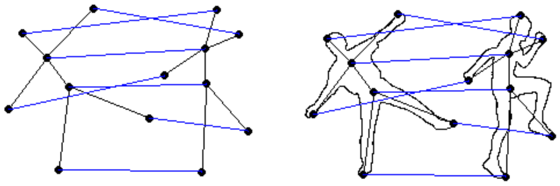
**Figure 4. (a), (b), (c) and (d) are examples of flower-shape contours and the numbers in the parentheses are their sizes. The degree of similarity is measured by the cost of best match while the quadruple includes the sizes of contours and the sizes of omitted contour segments of the best match due to cuts or merges. Example (e) is to demonstrate that our method is not a sequential one. The optimal match in (f) is derived with an $M$-operation between the two branches of $F_2$. The size of contour segments omitted, due to the merge, is $52$. In (g), a $C$-operation to remove the two branches (total size is $277$) is required to obtain a good matching.**



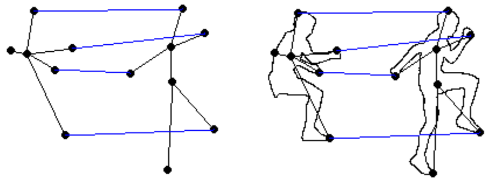
(a) Cost: $70.6646$ $(668, 714, 0, 0)$ (exact matching)



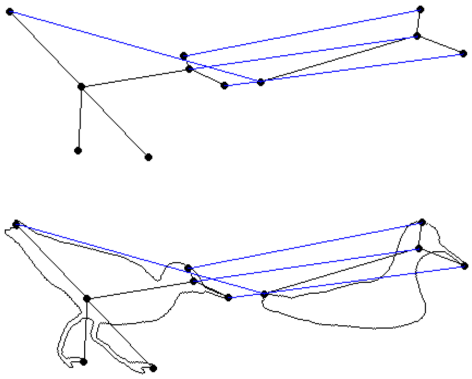
(b) Cost: $75.7528$ $(738, 711, 0, 0)$ (exact matching)



(c) Cost: $78.5166$ $(714, 738, 0, 0)$ (exact matching)



(d) Cost: $112.913$ $(501, 738, 0, 211)$ (cut)



(e) Cost: $109.784$ $(739, 477, 299, 0)$ (cut)

**Figure 5. Results (a), (b) and (c) are examples that the SA-tree shape comparison method can account for articulations and global shape information. To handle occlusions is also straightforward as shown in (c) and (d) with a $C$-operation.**