KMITL Sci. J. Vol. 5 No. 1 Feb. 2005

A SHAPE-MATCHING TECHNIQUE USING SKELETAL GRAPHS

Nualsawat Hiransakolwong

Mathematics and Computer Science department,
King Mongkut's Institute of Technology
Ladkrabang, Bangkok 10520 Thailand
E-mail: khnualsa@kmitl.ac.th

ABSTRACT

A novel shape-matching algorithm using skeletal graphs is proposed in this paper. The topology of skeletal graphs is captured and compared at the node level. Such graph representation allows preservation of the skeletal graph's coherence without sacrifying the flexibility of matching similar portions of graphs across different levels. Using appropriate sampling resolution, the proposed approach is able to achieve a high recognition rate, and at the same time, significantly reduce space and time complexity of matching. This approach is tested against the Directed Acyclic Graph (DAG) method on noisy graphs and occluded or cluttered scenes. The results show that this approach is an effective and efficient technique for shape recognition.

Keywords: Skeletal graph, graph matching, shape recognition, shock graph

1. INTRODUCTION

Although there has been extensive work in the area of shape representation and matching, shape recognition is still an open research problem. Many shape-matching approaches have emerged, but high space, time complexity and moderate recognition rate are still the limiting factors for their acceptance.

Some shape-recognition techniques are known as curve outline-based matching methods [3]. They often suffer from one or more of the following drawbacks: asymmetric treatment of the two curves, sensitivity to sampling, lack of rotation, scaling invariants and sensitivity to articulations and deformations of parts. Outline-based methods [4] have the advantage of not requiring ordered boundary points, but the match does not necessarily preserve the coherence of shapes in that the relationship among portions of shape in the process of matching may not be preserved.

Zhu and Yuille [5] have proposed a framework (FORMS) for matching animate shapes by comparing their skeletal graphs, the medial axis of shapes. However, the applicability to inanimate objects is limited due to the choice of primitives used in modeling. A variant of the medial axis is the shock structure, which is obtained by viewing the medial axis as the locus of singularities (shocks). Many approaches to shock graph matching have emerged. However, these approaches have not been extensively tested on noisy graphs, occluded scenes or cluttered scenes.

This paper proposes a technique that significantly reduces space and time requirements while improving recognition rate. This approach is based on many-to-many graph matching of skeletal graphs constructed from shapes. The matching algorithm is shown robust to scaling, rotation and translation. It is less sensitive to noise and occlusion. Experimental results show that this method is significantly better than a recently proposed alternative shock graph represented in the Directed Acyclic Graph (DAG) method.

2. SKELETAL GRAPHS MATCHING

A skeleton is an undirected tree captured from a silhouette of a binary image using the medial-axis method [6], see Figure 1. A skeleton tree consists of nodes and edges. There is a length associated with each edge. Each node has a maximum of eight connected neighbors. The angle between edges is a multiple of 45 degrees. The level of a node is defined as how close it is to the center of the tree, where a center is a vertex u such that the maximum value of the distances between u and all other vertices is a minimum. It is a well-known fact a tree has either a single center, or two centers connected via an edge [9]. A skeletal tree will be represented from the leaf nodes toward the center(s). The "root" of this representation corresponds to a center node. In the case of two centers, selecting either as root node will determine the level of all nodes in the skeleton.

In order to represent a skeletal graph, the topology of its individual nodes, i.e., the nodes' connected neighbors, the relationship with each neighbor and the length of the edges are capture. A signature table is constructed to keep information regarding the topology of nodes in the graph. There is only one table to store the nodes' information for the entire dataset.

The signature table consists of three attributes: signature, ID and "node detail". Thus, a row in the table is a 3-tuple. ID is a positive integer assigned to each signature. It serves as an identification of a node's topology. A signature consists of eight fields, corresponding to eight possible connected neighbors. Each field identifies the existence of a connected neighbor, the topology via an ID and the relationship with the node. A field containing 0 signifies there is no connected neighbor at that angle. A field of 3, for example, indicates there is a connected neighbor whose topological ID is 3, while a field of "P" implies the connected neighbor is its parent node. The fields are filled starting with P (the parent) or the highest ID for root nodes and going in the clockwise direction.

Node detail captures the fact that there could be many nodes with the same signature and ID. It is a linked list of nodes, each in the form *node/length/parent_node/image_number(list of children nodes or parent)*. The *length* is the length of the edge connecting the *node* with its *parent_node*.

The Table 1 shows the signature table filled with node information of the two example skeleton trees, Figures 1 (a) and (b). The first row encodes the information for all leaf nodes. They have signature (P,0,0,0,0,0,0,0), which is assigned ID = 1. Node detail is a linked list of all leaves in both trees in the order of processing. The next type of node, whose ID = 2, has signature (1,1,1,0,1,0,0,0). There is only one node, node 2 of Figure 1 (a), of this type. Therefore, Node detail contains 2/0/0/1a (3,4,5,1). The signature and the Node detail indicate that node 2 of Figure 1 (a) is the root node (length = 0, and no parent), whose children nodes are nodes 3, 4, 5 and 1, counting clockwise.

The last row of Table 1 describes node 3 of Figure 1 (b). This node is the root node and has three children nodes: 5, 2 and 4. Its signature indicates that its children node 5 is a node of ID = 4, node 2 of ID = 3 and node 4 of ID = 1, see Figure 1 (b).

Signature	ID	Node detail
(P,0,0,0,0,0,0,0)	1	1/74/2/1a (2), 3/68/2/1a (2), 4/37/2/1a (2), 5/23/2/1a (2), 1/28/2/1b (2), 4/75/3/1b (3), 6/60/5/1b (5), 7/62/5/1b (5), 8/60/2/1b (2)
(1,1,1,0,1,0,0,0)	2	2/0/0/1a (3,4,5,1)
(P,0,1,0,0,0,1,0)	3	2/14/3/1b (3,8,1)
(P,0,1,0,1,0,0,0)	4	5/22/3/1b (3,6,7)
(4,0,3,0,0,0,1,0)	5	3/0/0/1b (5,2,4)

Table 1: The signature table for Figures 1 (a) and (b)

2.1 CONSTRUCTION OF THE SIGNATURE TABLE

Node's signature is coded from leaves to root. The procedure to construct the signature table is given below. For each skeletal graph:

1. While the skeleton tree is not empty,
2. Code all leaf nodes.
3. Temporarily store the configuration of the parent nodes of the leaves (e.g., positions).
4. Delete all leaves from the skeleton tree.
5. Go to step 1.

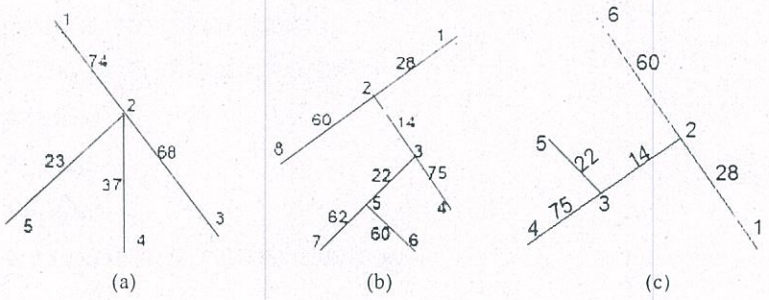

Figure 1: (a) and (b) two DB skeleton trees and (c) a query skeleton tree

Figure 1(c) shows a query skeleton. This is the 2-root skeleton, (i.e., there are 2 centers, nodes 2 and 3); therefore, two different signature tables are possible, one corresponding to each root node. Table 2 is the signature table for figure 1(c) with node 3 as root. A new ID, 6, is created because a new topology was detected.

Signature	ID	Node detail
(P,0,0,0,0,0,0,0)	1	1/28/2/1c (2), 4/75/3/1c (3), 5/22/3/1c (3), 6/60/2/1c (2)
(P,0,1,0,0,0,1,0)	3	2/14/3/1c (3,6,1)
(3,0,0,0,1,0,1,0)	6	3/0/0/1c (2,4,5)

Table 2: The signature table for Figure 1(c) with node 3 as root.

2.2 MATCHING SHAPES

The idea behind this shape recognition is as follows. When a skeleton is a 2-root type, there are two possible configurations (Figure 1 (c)). To ensure matching, e.g., between a 2-root query and a 2-root skeleton, both configurations of either need to be compared. One configuration for each database skeleton, either 1-root or 2-root, is coded. 2-root queries will be coded in both configurations. Signatures are used to identify two types of matches: an exact match for two nodes of the same topology and at the same level, and a close match for two nodes of similar topologies at any levels. Exact matches can be declared when the two signatures are the same.

Close matches are ones when query node's children match some children of the DB node. For example, query signature (P,0,0,0,1,0,1,0) is a close match with the DB signature (P,0,1,0,1,0,1,0). When the query node is a root node, the DB signature is compared with all circular-shiftings of the query signature. For example, query's signature (3,0,0,0,1,0,1,0) is shifted to (1,0,3,0,0,0,1,0), where a close match with DB signature (4,0,3,0,0,0,1,0) is declared. For a close match, the order of the children in "node detail" is updated to reflect the change in its signature. The matching algorithm is summarized as follows:

1. Clear all marks on the DB signatures.
2. While there are query signatures, do
   i. Select one query signature,
   ii. Identify all DB exact and close matched signatures,
   iii. Mark all DB matches accordingly.
   iv. Remove the query signature.

To filter out unlikely (close) matches, (DB *node_length*)/(query *node_length*) ratios and their median are computed. Matches whose ratio is outside the range [*median ratio – threshold, median ratio + threshold*] are eliminated. For each of the remaining matching pairs, exact or close match, the matching score is computed:

$$\text{Score} = w\left(\frac{CN(T_1,T_2)}{\max(|T_1|,|T_2|)}\right) + (1\text{-}w)\left(\frac{CP(T_1,T_2)+1}{\max(|T_1|,|T_2|)}\right)$$

where *CN* is the number of matched nodes, *CP* the number of matched nodes whose ratios in the range [*median ratio–threshold, median ratio+ threshold*]. $|T_1|$ is the size of query skeleton tree T_1, and $|T_2|$ is of DB skeleton tree T_2. The w is the weight in [0,1], which is set to adjust the relative significance of the two terms. In experiments, $w = 0.5$ and *threshold* = 0.25.

2.3 PROPERTIES OF THE ALGORITHM

To prove the properties of the algorithm, the distance measure equivalent is defined to the scoring function:

$$d(T_1, T_2) = 1 - \text{score} = 1 - w\frac{CN(T_1,T_2)}{\max(|T_1|,|T_2|)} - (1-w)\frac{CP(T_1,T_2)+1}{\max(|T_1|,|T_2|)}$$

Theorem 1: For any skeleton trees T_1, T_2 and T_3 the following properties hold true. (See proof at [8].)

i. $0 \le d(T_1, T_2) \le 1$

ii. $d(T_1, T_2) = 0 \Leftrightarrow T_1$ is *isomorphic to* T_2

iii. $d(T_1, T_2) = d(T_2, T_1)$

iv. $d(T_1, T_3) \le d(T_1, T_2) + d(T_2, T_3)$

Theorem 1 indicates that the measure ($d(T_1, T_2)$) is a distance metric. The matching procedure based on node topology is translation and rotation invariant, and robust to scaling. Let n be the number of the nodes of skeletal graphs and m the number of rows (signatures) in the signature table. The following theorem can also be proved (see proof at [8]).

Theorem 2: The space complexity of the proposed technique is O(n) for each query, and the time complexity is O(mn) for executing a query against the entire database, where m = the number of rows in the signature table, and n = the number of nodes in the query's skeletal graph.

Compared with the DAG method, the proposed method is a significant improvement. The DAG method takes $O(n^3M)$ for matching time, where M is the number of database images. In experiments, there were about m=1,700 signatures for a total of M=4,000 database images. In terms of storage space, DAG uses eigenspaces of the adjacent matrix, costing $O(n^2)$ for each image.

3. EXPERIMENT STUDY

3.1 DATASETS AND METRICS

The dataset comes from [7] and some images are added to test various features of the proposed technique. Images are views computed from 3-D graphics models. Each is centered in a uniformly tessellated view sphere and a silhouette is generated for each vertex in the tessellation. A skeletal graph is computed for each silhouette with a particular block size. The depth first search algorithm is used to traverse the graph. A node is marked where the graph changes direction. There are 147 queries, which can be occluded or unoccluded images.

3.2 CONFIGURATION

Since the block size greatly affects both processing time and recognition rate, the best block size is determined to construct skeleton trees. Experiments are tested with five different block sizes: 1x1, 2x2, 4x4, 8x8 and 16x16. Respectively, the processing times for these blocks are 28, 17, 11.29, 8.46, and 8 minutes for 4,000 images. The relationship between recognition rates and block size is more interesting; see Figure 2. When block size is increase from the pixel level to 2x2 to 4x4, the recognition rate improves. This indicates that block size 4x4 helps remove noise and smooth out the skeletal graphs. Recognition rates, however, deteriorate rapidly for larger block sizes. This is attributed to the fact that the skeletal graphs now become too coarse to capture shapes in sufficient details. Since block size 4x4 yields the highest recognition rate at reasonable processing time. This block size 4x4 is used for the proposed technique in subsequent experiments.

3.3 COMPARATIVE STUDY

In this experiment, the proposed technique is evaluated its performance against with the DAG on a set of unoccluded queries. The same dataset as the DAG's, consisting of up to 1,408 views of 11 objects, is used and each object having 128 views. The results are summarized below (see Figure 3 and [8]):

- The proposed technique's recognition rate is consistently better than the DAG's as the database size increases.
- For unoccluded queries the proposed method achieves higher recognition rate faster than the DAG does, as the number of views per object increases.
- For occluded queries, the recognition rates comparably reduce as the percentage of occlusion grows. Both methods yield recognition of greater than 70, even for 50% occluded objects.

Figures 4 to 7 show the results of some example queries against database of about 4,000 images. For each query, top-ranked images are displayed along with their scores. The higher the score is, the closer is the match. Table 3 summarizes the results of these queries.

Observe that the ranks of relevant images are very high, and the images are views at various angles and scaling of the queried object. In the last query, see Figure 7, for heavily occluded queried objects (dog's legs are occluded), the proposed approach is still effective to achieve high recall (compared with the recall in the previous query: "unoccluded dog", see Table 3).

Figure	Query	Relevant	Retrieved	Recall
4	Human	63	58	92%
5	Dolphin	52	46	88%
6	Dog	28	26	92.8%
7	Occluded dog	28	26	92.8%

Table 3 Summary of the results of these queries

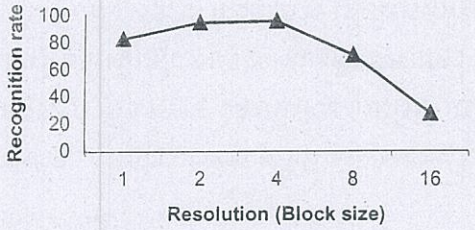

Figure 2: Recognition performance with various resolutions

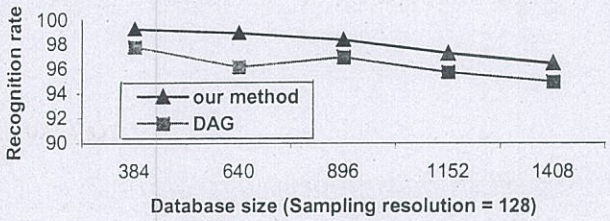

Figure 3: Recognition performance with various database sizes

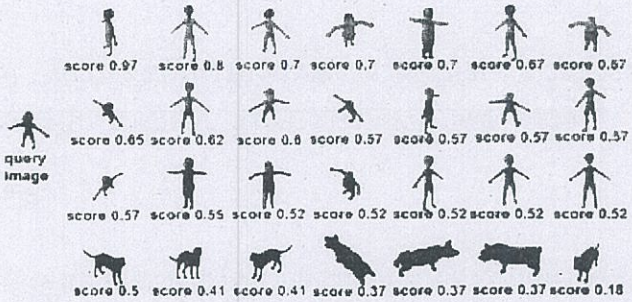

Figure 4: Query image and top-ranked images

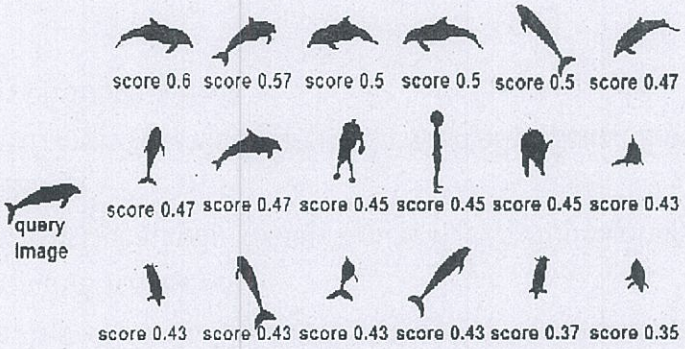

Figure 5: Query image and top-ranked images

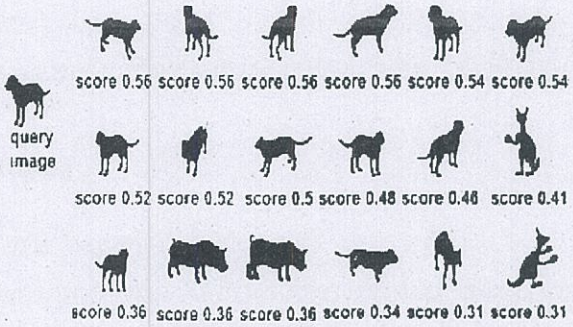

Figure 6: "Unoccluded dog" and top-ranked images

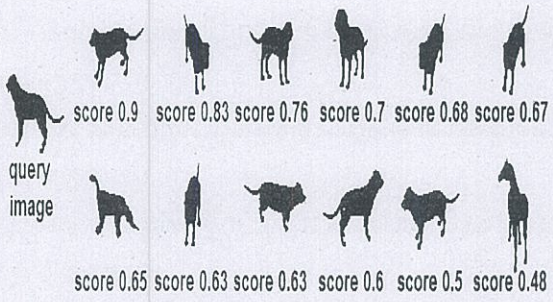

Figure 7: "Occluded dog" and top-ranked images

4. CONCLUSIONS

Shape matching is an important yet open problem. This paper proposed a matching algorithm for skeletal graphs, which are used to represent shapes. The topology of skeletal graphs is captured and compared at the node level. The present technique achieves a high recognition rate, and at the same time, significantly reduces space and time complexity of matching. The results show that the proposed approach is an effective and efficient technique for shape recognition.

5. REFERENCE

[1] D. Macrini, A. Shokoufandeh, S. Dickinson, K. Siddiqi and S. Zucker. View-Based 3-D Object Recognition using Shock Graphs. *ICPR*, 2002.

[2] T. B. Sebastian, P. N. Klein and B. B. Kimia. Recognition of Shapes by Editing Shock Graphs. *ICCV*, pages 755-762, 2001.

[3] T. B. Sebastian, P. N. Klein and B. B. Kimia. Alignment based recognition of shape outlines. *IWVF*, pages 606-618, 2001.

[4] S. Belongie and J. Malik. Matching with shape contexts. *CBAIVL*, 2000.

[5] S. C. Zhu znd A. L. Yuille. FORMS: A flexible object recognition and modeling system. *IJCV*, 20(3), 1996.

[6] H. Blum. *A Transformation for Extracting New Descriptors of shape*, pages 362-380, MIT Press, 1967.

[7] http://www.cs.toronto.edu/~dmac/download.html

[8] http://www.cs.ucf.edu/~nual/proof.doc

[9] F. Harary. *Graph Theory*, Addison-Wesley, Reading, Massachusetts, page 35, 1969.