Skeleton Based Shape Matching and Retrieval

H. Sundar [*] Rutgers University
D. Silver [†] Rutgers University
N. Gagvani [‡] Sarnoff Corporation
S. Dickinson [§] University of Toronto

Abstract

In this paper, we describe a novel method for searching and comparing 3D objects. The method encodes the geometric and topological information in the form of a skeletal graph and uses graph matching techniques to match the skeletons and to compare them. The skeletal graphs can be manually annotated to refine or restructure the search. This helps in choosing between a topological similarity and a geometric (shape) similarity. A feature of skeletal matching is the ability to perform part-matching, and its inherent intuitiveness, which helps in defining the search and in visualizing the results. Also, the matching results, which are presented in a per-node basis can be used for driving a number of registration algorithms, most of which require a good initial guess to perform registration. In this paper, we also describe a visualization tool to aid in the selection and specification of the matched objects.

1 Introduction

With the development of advanced 3D scanning devices, the ubiquity of VRML and the web, and the rise in computational processing power, the number of 3D models available is increasing rapidly. The 3D models include both polygonal datasets (such as VRML files, Autocad, etc.) and volumetric files (3D datasets). Unfortunately, and very much like image files, these 3D models are not easily searchable. The problem of matching 3D shapes is a difficult one and has been the recent focus of research efforts. Text descriptors of 3D shapes can also be used for searching as is done for images [16, 25]. However, in general this is not the case and a matching algorithm would be useful to both *retrieve and compare* 3D models from a database of models. It is also useful in scientific applications where one may want to compare and visualize similar models.

While the 3D matching problem is related to the shape matching problem in computer vision (detecting a 3D shape from 2D images), there are some significant differences. 3D models generally do not have any of the occlusion problems associated with sensory images. Furthermore, image acquisition effects, such as camera position/viewing angle, lights, etc. are not a factor in 3D. However, many 3D models are degenerate [24] and contain missing, intersecting or disjoint polygons. Furthermore, because of the added dimension and size, 3D models are difficult to compare and how they may match is not necessarily obvious.

In general, matching consists of the problem of quickly finding promising candidates from the database given a query and the problem of checking each of these candidates using a full-blown matching operation (verification). Computing the *best match* is a combination of both steps, and is a relative concept depending upon the application, i.e. it can mean anything from classification into categories to actual shape similarity. Our approach is to use a matching methodology based upon a "skeleton" of a 3D shape. The skeleton used in this context is a graph-like representation of a 3D object. This *skeletal graph* is computed directly from the 3D object and contains the topological information about the 3D object in terms of the graph and local shape descriptors, which are held at each node in the graph. These local shape descriptors contain information to aid the matching program. This information could include the mean, radius, degrees of freedom about the joint (for topological matching), or the degree of importance of a particular joint or node. The skeleton is a nice shape descriptor because it captures the notion of parts and components. It is also an intuitive shape descriptor and so can be edited by the user to help refine a particular search query and understand the match.

In this paper, we present a methodology, that takes 3D objects, both volumetric and polygonal, and stores the shape information as an indexed database of skeletal graphs. We store along with the graph, the *topological signature vector* of the graph, which is a low-dimensional index that captures both the local and global structural properties. The database can then be queried to determine the best matches for a given 3D shape. The search is fast as the graphs are indexed. We follow this up by performing a graph match-

[*] hsundar@caip.rutgers.edu
[†] silver@caip.rutgers.edu
[‡] ngagvani@sarnoff.com
[§] sven@cs.toronto.edu

ing on the top responses from our query on the indexed database. In the next section, a short literature review is presented followed by an overview of the skeletonization, indexing and matching algorithm. The results of the matching algorithm and visualization of the results is presented in Sections 6 and 7.

2 Previous Work

Shape matching is one of the fundamental problems in computer vision and a full treatment of the subject is beyond the scope of this paper. Below, we concentrate on research areas related to the efforts in this paper including 3D object matching, volumetric matching and 3D registration. In [24], a 3D shape matching algorithm and system is presented which computes a *shape signature* for each object and then matches an object into a database of signatures. The shape functions are computed stochastically by randomly sampling over the shape and then creating a continuous probability distribution as a signature for the 3D shape. Various shape functions were utilized, such as the angle between 3 random points on the surface, the distance between two random points, etc. The advantage of this method is that no complex feature extraction is necessary, and there are well-defined methods for computing the similarity of two distributions. The resulting framework is an effective technique for classifying objects, and is robust to small perturbations on the object boundary. However, such distributions do not capture where features are located and can not deal with partial queries, missing features, or part articulation.

In [13], an interactive algorithm is presented which employs a mechanism called "relevance feedback" to iteratively refine a search according to user preferences [11]. The notion of relevance feedback can be applied to any database indexing scheme, provided that there is a mapping between user preferences and the components of the distance function used to rank objects based on similarity. This paper focuses more on the relevance feedback, adopting a simple object model using moments (which require the object to be oriented and scaled canonically). Moments are a global property, and hence cannot accommodate missing or partial information (in the query), nor can they accommodate object part articulation. However, the notion of relevance feedback is an important one in 3D.

In [18], a 3D matching algorithm is presented based upon a multiresolution Reeb graph (MRG). The MRG is computed for each 3D shape and then a graph matching technique is used to match the MRG's. The computed feature, in this case the geodesic distance to all other points, is partitioned into intervals which, in turn, give rise to a Reeb graph. For 3D objects, the geodesic distance is invariant to part articulation, although not to missing or partial data. Problems can arise in the mapping from the continuous geodesic function to the discrete graph as the structure or topology of the graph is entirely dependent on the choice of interval size. The nodes in the graph have no intuitive interpretation with respect to parts of the object. Geometric distortions of the parts relative to each other (or significant within part distortions) could therefore have significant impact on the graph structure. The matching algorithm is coarse-to-fine, and resembles a number of other approaches in the vision community [34].

In [4], results are presented to "match" volumes based upon a set of points chosen within the volumes. This is a specific technique used to orient two volumes. Also related is 3D registration using skeletons (See for example [5, 20, 3, 10, 1]). These techniques are not general purpose matching algorithms and assume the skeletons to be registered are approximately the same. However, the ideas of super-imposition and co-registration can be used as part of a general matching program that uses skeletonization. For example, once a match is determined, it is very helpful to see how the two volumes are matched and aligned, or if a part-match was determined, which parts of the volume matched. Furthermore, most registration algorithms need an initial guess to begin with. The work described in this paper provides that initial super-imposition as well as providing an indexing and retrieval methodology.

3 Overview of Skeleton Matching

An overview of the skeleton matching process is given in Figure 1. The steps in this process include: obtaining a volume, computing a set of skeletal nodes, connecting the nodes into a graph, and then indexing into a database and/or verification with one of more objects. The results of the match are then visualized. In this paper, we focus on the construction of the skeleton and preliminary results of using the graph matching in conjunction with skeletonization. The indexing is similar to that described in [34] and is done by assigning to each non-terminal node a vector representing the eigenvectors of the subgraph adjacency matrix rooted at that node. This is used to query the database in a fast way, before the actual matching is performed on a per-node basis.

The skeleton is a nice shape descriptor because it can be utilized in the following ways:

- Part/Component Matching: In contrast to a global shape measure, skeleton-matching can accommodate part-matching, i.e. whether the object to be matched can be found as part of a larger object, or vice versa. This feature can potentially give the user flexibility towards the matching algorithm, allowing them to specify what part of the object they would like to match

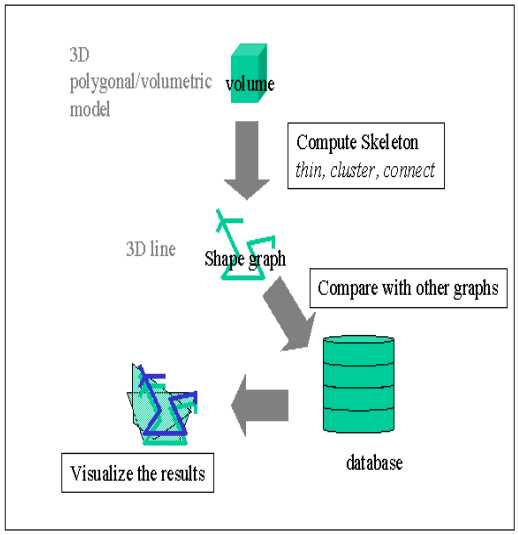

Figure 1. Overview of the skeleton based matching/comparison algorithm

or whether the matching algorithm should weight one part of the object more than another.

- Visualization: The skeleton can be used to register one object to another and visualize the result. This is very important in scientific applications where one is interested in both finding a similar object and understanding the extent of the similarity.
- Intuitiveness: The skeleton is an intuitive representation of shape and can be understood by the user, allowing the user more control in the matching process.
- Articulation: The method presented here can be used for articulated object matching, because the skeleton topology does not change during articulated motion.
- Indexing: We can index the skeletal graph for restricting the search space for the graph matching process.

3.1 3D Skeletonization

The first step in the skeleton-based 3D matching is computing the skeleton.The term *skeleton* has many meanings. It generally refers to a "central-spine" or "stick-figure" like representation of an object. The line is centered within the 3D/2D object. For 2D objects, the skeleton is related to the medial-axis [7] of the 2D picture. For 3D objects a medial *surface* is computed (see [21, 9]). To use graph matching [31] what is needed is a medial core/skeleton [1, 6, 14, 17, 28, 19] also known as a curve-skeleton [36] which can be represented as a graph. The method we utilize here is a parameter-based thinning algorithm described in [14]. (A full survey of skeletal methods is also presented in [14]). It is similar in principle to [9, 28] but thresholds on the average of the skeletal voxels instead of computing the gradient first. All of the methods must be further thresholded and clustered to reduced the surfaces to a manageable graph. The advantage of using the method in [14] is that a graph-like skeleton of desired thickness can be computed very quickly for large 3D datasets.

This algorithm thins the volumes to a desired threshold based upon a parameter given by the user. A family of different point sets results, each one thinner than its parent. This point set, termed *skeletal voxels* is unconnected, and must be connected to form an appropriate stick-figure representation. In what follows, we describe the various steps necessary to compute the skeleton/graph representation.

3.2 Volumetric Thinning

To thin the volume, the distance field of each voxel in the object is computed. In the discussion which follows, we use a volumetric object which has been segmented from the background. For any matching applications object boundaries tend to be well defined. For polygonal models, a voxelizer program [12] is first used to convert the polygonal model to a volume model.

The distance transform [26] at an object voxel is the minimum distance from the voxel to the boundary of the volumetric object. Various metrics can be used to compute the distance transform, such as a quasi-Euclidean [8] or a true Euclidean metric [27]. In the discussion which follows, the Euclidean metric is used. The distance field or distance transform value (DT) of a voxel is the radius of a sphere centered at that voxel. Such a sphere will be tangential to the boundary of the object. If we fill in the sphere, we can reconstruct a part of the object touching the boundary. For 2D images, Niblack, et al [22] have described an algorithm which identifies the set of pixels, the distance transform of which is sufficient to completely reconstruct the object by constructing discrete disks around those pixels. Nilsson and Danielsson [23] present an algorithm to compute the minimal set of such pixels. In general, for objects with complex boundaries, these methods yield far too many disks if complete coverage is desired. This problem is compounded for volumetric objects. The number of voxels whose spheres will exactly cover every boundary voxel, while much smaller than the original number of voxels, is too large for interactive manipulation.

In [14], we described a thinning technique using a thinness parameter. This method classifies voxels based on their importance for boundary coverage. We use a heuristic which compares the distance transform at a voxel with

that of its 26-neighbors. If the distance transform at a voxel is much greater than those of its 26-neighbors, the sphere centered at that voxel is likely to include most or all of the spheres centered at its neighboring voxels. A thinness parameter determines how much larger such a sphere should be for its center voxel to be considered important for boundary coverage. For every voxel p, we compute MNT_p, the mean of the distance transform of its 26-neighbors. The thinness parameter TP is then the difference of DT_p, the distance transform at voxel p and MNT_p, the mean distance transform of its neighbors. It is summarized in the equation below.

Equation 1 $TP = DT_p - MNT_p$, *for every voxel* p *in the object, where* $MNT_p = \frac{\sum_{i=1}^{26} DT_{q_i}}{26}, p, q_i \in object - voxels, q_i$ *is a 26-neighbor of* p.

A low value of TP indicates that p is important for boundary coverage if its distance transform is slightly greater than that of its neighbors. A high value of TP means that p must have a distance transform that is much greater than that of its neighbors. Consequently, as TP is decreased from infinity, the number of spheres covering the object increase. A more theoretical description for the thinness parameter can be found in [14]. The thinness parameter allows us to represent a volumetric object at several levels of detail. For some value of the thinness parameter, we can rapidly extract voxels whose distance transform is greater than those of their neighbors by at least TP. Construction of the spheres centered at those voxels yields a representation of the boundary of the object. We call the set of these spheres the *reconstruction-spheres*, because they can be used to reconstruct the boundary of the object (at that thinness value) [15].

Using different levels of the thinness parameter in **Equation 1** allows us to create bounding volumes at different levels of detail. With a higher thinness parameter, we get a larger bounding volume but fewer spheres, while a lower thinness parameter yields a tighter bounding volume with more spheres. We compute $DT - MNT$, the difference between the distance transform and the mean of the neighbors' distance transform for every voxel. Voxels are sorted in decreasing order of $DT - MNT$. For a desired number of voxels, n, at some level of description, we extract the first n voxels from this sorted list.

Each point in the volume is classified based upon how much its distance-field value is larger than its neighbors. Varying this value results in a family of skeletons. An example is shown in Figure 2 below. Note when a low TP is used (i.e. those voxels that are not much greater than their neighbors), the points lie on the medial surface planes. As the TP is increased, the points left are those that lie along the medial axis of the individual planes. The advantage of computing the skeleton using this methodology is its simplicity and speed.

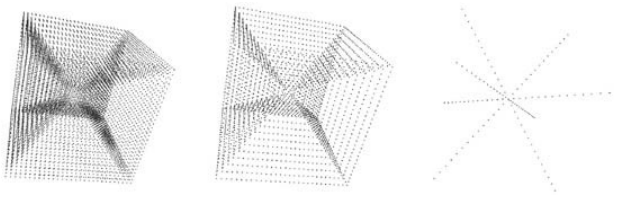

Figure 2. Thinning a volumetric cube into a skeletal-graph. The left image shows a volumetric cube. The middle image is thinned into a set of points lying on the medial-axis planes. The rightmost image contains the line-like figure along the medial axis of the planes.

3.3 Clustering

The above thinning process was mainly developed for reconstruct-ability and to support volume animation [15]. Since that is not a criterion here, we can utilize a clustering algorithm to further cluster the thinned voxels, this helps lessen the effect of many small perturbations on the surface and reduces the number of nodes necessary for skeletal graph construction. However, the original skeletal values are saved and used during the graph matching process for shape calculation (see below). The clustering algorithm is distance based. We add a point $p(x, y, z)$ to a cluster C_a if

$$dist(p(x, y, z), p(i, j, k)) < D_{threshold} \ \& \ NP(p(x, y, z), p(i, j, k))$$

for all $p(i, j, k) \in C_a$, where $dist(p_1, p_2)$ is the distance between two points and $NP(p_1, p_2)$ determines whether the surface is pierced since it is generally not desirable to cluster across surface boundaries even if the points are close (e.g. points at the fingertip of a hand should not be clustered together).

3.4 Generation of the Skeletal Graph

After thinning and clustering, the skeletal points are unconnected. To utilize the shape graph matching [31] the points have to be converted to a directed acyclic graph (DAG). We also have to ensure that the shape information is preserved during this process and that the method is tolerant enough so that minor changes in the position of skeletal points do not produce drastically different shape graphs.

We first generate an undirected acyclic shape graph out of the skeletal points, by applying the *Minimum Spanning*

Tree (MST) algorithm, with all the edges weighted proportional to their length (and or distance transform see [14]). Edges are restricted to be within the boundaries of the object. A directed graph is created by directing edges from the voxel with the higher distance transform to the one with lower distance transform. In principle, it is similar to the *shock graph* concept in [31] where larger features are directed towards smaller ones. The MST is sensitive to distance variation at the joints which could result in incorrect connectivity structure. The tolerance of the matching process accommodates these perturbations. In Section 8, we briefly describe a newer skeletonization method that does not use the MST to compute the skeleton.

Once a graph is produced it can be further simplified by collapsing straight edges. Example skeletal graphs are shown in Figure 3 and Figure 5. The complexity of the shape determines the size of the skeleton, and the size can be modified by the user by setting the appropriate thinness-parameter (TP). The skeletons in Figure 3 were all generated automatically using a middle TP value. For a range of TP values, a hierarchical set of graphs results [14], each one more dense than the next. Each node in the skeletal graph represents a segment in the original skeleton. This node carries information about the local shape of the segment in the form of a cloud of points, obtained from the volume thinning, and associated with that segment. Each edge in the skeletal graph corresponds to a joint in the original skeleton. Thus, each edge can potentially store information about the *flexibility* of a joint/junction in the 3D shape. Similarly a flexibility metric can be attached to the segments by having an extra parameter for each node. Each node in the graph also contains the TSV(topological signature vector), which is used for indexing.

If we have a DAG, T, whose maximum branching factor is ΔT, and let the subgraphs of its root be $T_1, T_2, ..., T_{\delta(T)}$. For each subgraph, T_i, whose root degree is $\delta(T_i)$, we compute the magnitudes of the eigenvalues of T_i's submatrix, sort them in decreasing order by absolute value, and let S_i be the sum of the $\delta(T_i) - 1$ largest absolute values. The sorted S_i's become the components of a $\Delta(T)$-dimensional vector assigned to the DAG's root. If the number of S_i's is less than $\Delta(T)$, then the vector is padded with zeroes. We can recursively repeat this procedure, assigning a vector to each non-terminal node in the DAG, computed over the subgraph rooted at that node. This vector is the TSV.

4 Matching the Shape Graphs

In [34, 33, 30, 31], a matching algorithm is given which matches 2D shock graphs (rooted trees). At each node in the graph, a structural "signature" is defined, which characterizes the node's underlying subgraph structure. This signature is a low-dimensional vector whose components are

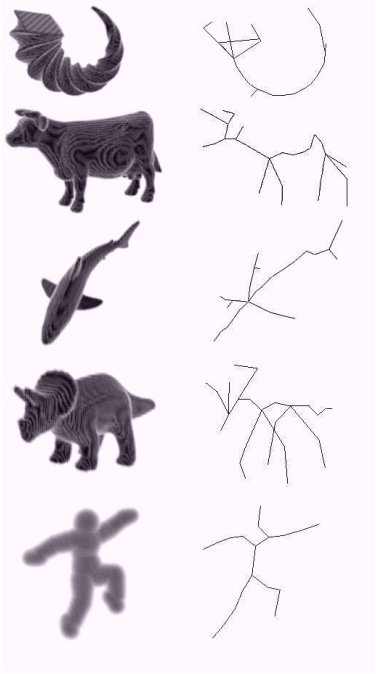

Figure 3. Sample skeletal graphs:In the left column, different volumes are shown. In the right are the resulting skeletal graphs.

based on the eigenvalues of the subgraph's $(0, 1)$ adjacency matrix. The eigenvalues of a graph (spectral graph theory) carry important structural information about the graph and possess important stability properties. Specifically, small perturbations in graph structure due to noise or minor shape perturbation will have a correspondingly small effect on the eigenvalues.

Matching two graphs is typically formulated as a largest isomorphic subgraph problem, whose complexity is prohibitive. Since contextual graph structure is effectively encoded in a node's signature vector, we could throw away all the edges in the graph and reformulate the problem as finding the maximum cardinality, minimum weight matching in a bipartite graph. In such a formulation, there is an edge between each node in one graph and each node in the other, whose edge weight represents the distance between the two nodes' structural signature vectors. Solving for a maximum cardinality, minimum weight matching involves choosing a subset of the edges in the bipartite graph which provide a one-to-one mapping, whose sum edge weights (distance) is small, and whose cardinality (matching size) is high.

Unfortunately, the above formulation does not yield a solution that is guaranteed to obey the hierarchical structure of the two graphs. Specifically, there's nothing to prevent nodes $n_1 1$ and $n_1 2$ in one graph, with $n_1 1$ an ancestor of

$n_1 2$, matching nodes $n_2 1$ and $n_2 2$, respectively, in the second graph, with $n_2 2$ an ancestor of $n_2 1$. To preserve the hierarchical (ancestral) relationships in the graph, we have combined a greedy form of the above bipartite formulation with a recursive, depth-first search. The approach can be thought of as a coarse to fine strategy, in which matching at higher levels of the tree is used to constrain matching at lower levels. The technique's complexity is better than $O(n^3)$, and its efficacy has been demonstrated in the presence of noise and occlusion (for 2D). Details can be found in [30, 33, 34, 31], which is inspired by the algorithm for matching graphs by Siddiqi et al [34]. The algorithm recursively finds matches between the trees, starting at the root of the tree, and proceeding down through the subtrees in a depth-first fashion. The algorithm has the ability to match two trees in the presence of noise (random insertions and deletions of nodes in the subtrees).

Two factors determine whether two nodes of the trees get matched: the first is a measure of the topological similarity of the subtrees rooted at the nodes, while the second is a measure of the local shape information at that node (about the edges in the skeletal graph). Although each skeleton only has a limited number of nodes, the radial distribution about each edge is preserved for matching. The radial distribution is represented by a more dense graph with the distance field values at each node. The dense graph is used to test for the radial distribution but not for the topological graph matching. For matching, either averages, max and min, or the actual distribution can be used depending upon the type of match desired. The values can also be normalized to account for scaling. This feature can be turned off if only topological similarities are desired.

The graph matching algorithm outputs a number of parameters that can be used to determine the "goodness" of the match. These include: the number of nodes matched (or the percentage of nodes matched from the target and from the key), the sizes of the clusters of nodes matched, and a detailed specification of which nodes were matched to which other nodes. A goodness measure is given for each node that is matched as well stating how accurate the match was. This is a factor of the various parameters given above. Because part matching is supported the percentage of nodes matched will indicate whether a part match has been found (and model nodes can be penalized if part matching is not desired). Note that the number of nodes in a graph is model dependent and is a function of the complexity of the object. Recent efforts [32] have also characterized ways in which geometric relations between the nodes in a graph can play a role in the matcher, parameterizing things like articulation, relative size, relative orientation, etc. of the parts on a node-by-node basis.

5 Interface

The interface to the matching program was written in Vtk. It consists of a key window, which can compute and display the skeleton at different resolutions, and options for computing a match with a particular set of skeletons. Results are displayed quantifying the match (or matches)using the parameters described previously. A snapshot of the interface is displayed in Figure 4. The skeleton can be automatically computed, or the user can modify the resolution of the skeleton using the slider bar. The automated skeletal computation computes the skeleton using a 50% TP value.

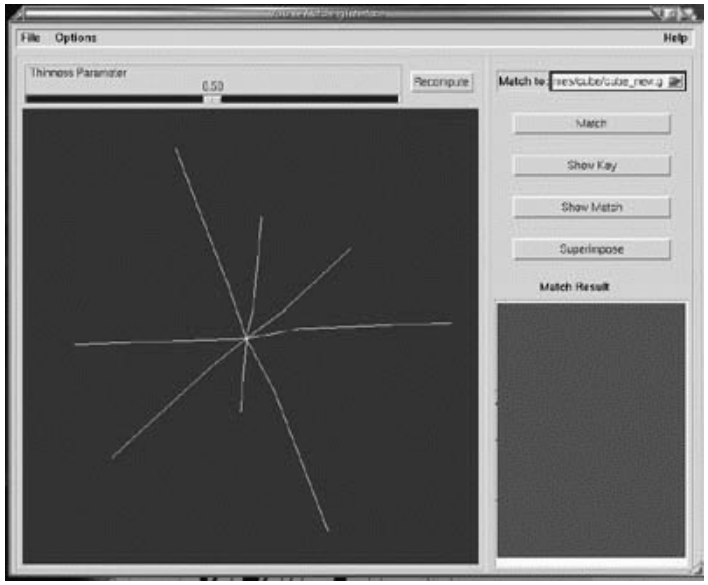

Figure 4. Matcher Interface and Visualizer

6 Results and Visualization

For the initial testing, a database of 100 different volumetric models was used (plus parts and articulations of models). Some of the models were originally polygonal datasets which were voxelized, while others were volumetric. We used the database to retrieve close matches based on the shape index, and also to quantify the difference based on the shape graphs.

To visualize the resulting match, either the skeletons or models can be visualized separately, or the results can be superimposed using the skeletons as a guide. Since the graph-matching algorithm outputs node-to-node correspondences, this information can be used to align the skeletons and the models. In Figure 5, an example of node-to-node correspondence is shown using two fish from the database (the best match for each from the database). In the left part of the image, the two models are shown aligned, and in the

right the skeletal graphs are shown. The node-to-node correspondence of the graphs are indicated by the colored vertices in the graphs (the red fish is the graph with red edges, the blue fish is the one with blue edges.) The first couple of nodes are used to align the skeletons (working down from the root). In the left window, the polygonal models are superimposed based upon the transformations applied to the skeletons. While aligning along a couple of nodes will most likely result in a good fit, a more accurate alignment can be calculated using a least squares approximation or other registration techniques.

An example of part matching is shown in Figure 6. The hand and the foot were matched to the entire body. Both the hand and foot were skeletonized separately and resulted in different skeletal-graph resolutions. In Figure 6 the matching nodes were used to align the graphs correctly. On the entire database match, the left hand also matched to the right hand, as did the corresponding feet. (Note the volumetric model of the foot had fused toes resulting in the fused skeleton.)

In Figure 7, two matches are shown. Both were the highest match computed in the database for that object. The associated skeletons are also shown with the node-to-node correspondence based upon both the topology and radial distance about the edge. Note how the tail of the dragon is unmatched. In Figure 8, an example of articulated matching is shown. Three different skeletons in a running sequence are shown with node-to-node correspondence.

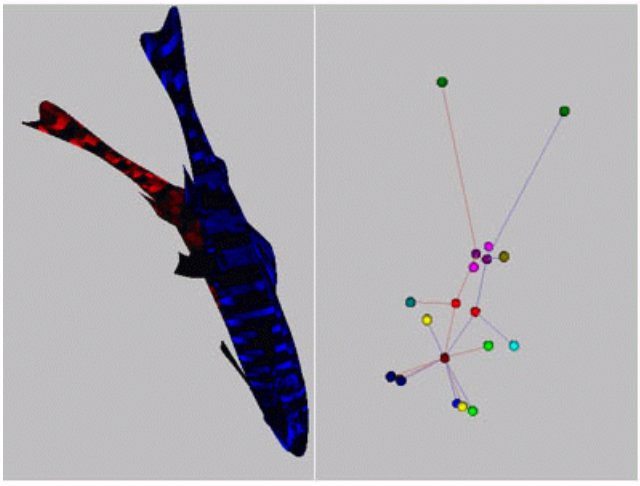

Figure 5. Visualizing the resulting matches. The left image shows the original polygonal models superimposed. The right image shows the skeletons. The nodes that are matched are color coded.

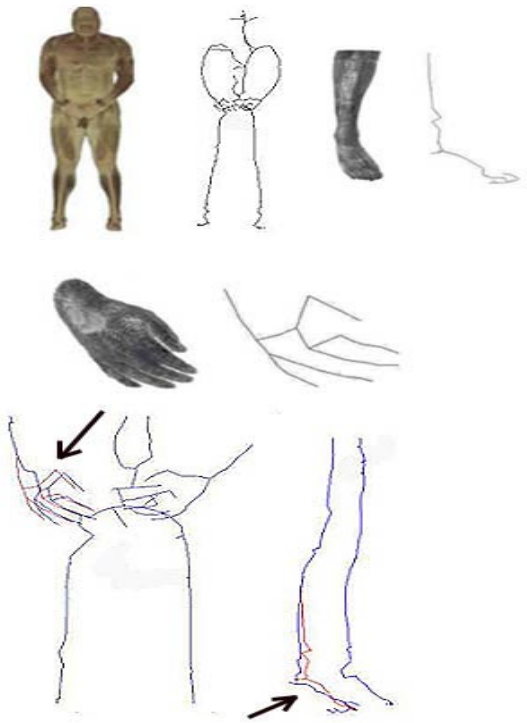

Figure 6. Part Matching. The skeletons of the Visible Man Dataset, and a foot and hand are shown. (The foot and hand were skeletonized separately.) The hand and the foot were matched to the entire body and the nodes which matched are shown with the two graphs superimposed (red graph belongs to the part).

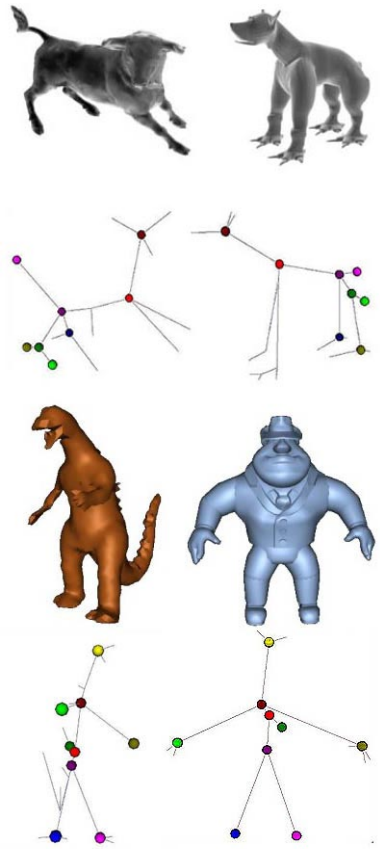

Figure 7. Individual matches in the database. The top images contains two different animals. The bottom a man and an animal. The node-to-node matches of the skeletons are also shown.

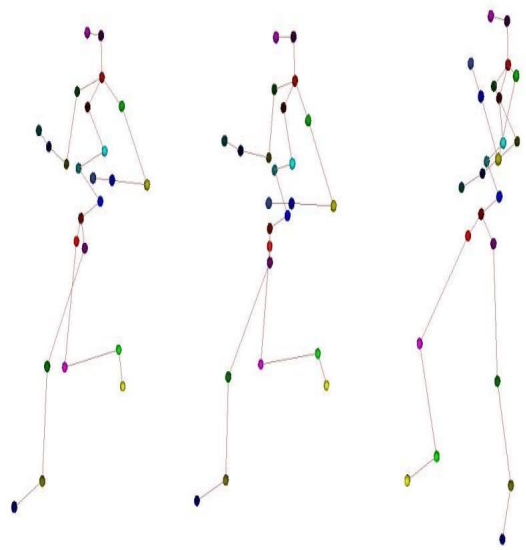

Figure 8. Articulated matching. The skeletons of an animated figure are matched.

7 Discussion

Our goal in this paper was to show the feasibility of using skeletons for shape matching and shape retrieval. While a skeleton is not a good descriptor of shape for all 3D objects (e.g. a bookcase), for many it does provide an accurate representation. While skeletal representations are good for part matching, one difficulty is that simple objects will match with smaller regions of complex objects. For instance, a small cylinder will match with other small cylinders and other models with cylindrical features. The tradeoff between part matching and skeletal complexity is task dependent and the algorithm can be modified based upon user preference. A hierarchical skeleton at different resolution would also allow the user more freedom in the type of shape desired.

We also note that the skeletal graph can be affected by the voxelization resolution and technique. For detailed polygonal objects, the fidelity of the resulting volume is clearly a factor of the resolution used for voxelization. Care must be exercised to avoid fusing finer features and thereby changing the topology of the graph. On the other hand, voxelization can also smooth out surface details, resulting in smoother skeletons. In this implementation, the resolution of the voxelization was related to the size of the polygonal model and the number of polygons. None of this is of concern if original voxel models are used (as in Figure 6). There is ongoing research investigating the computation of medial-axis and skeletal-graphs from polygonal

models [21, 1, 19]. These could then be used directly without the need to voxelize the models.

While any skeletonization process is generally sensitive to noise on the boundary (slight movements of voxels, or the addition of bumps and breaks) [37, 29], the overall impact of noise is reduced because of the clustering and thinness parameter. This can be seen in Figure 9 where noise was added to about 15% of the voxels on the boundary. Since the graph matching algorithm can match in the presence of noise (random insertions and deletions), small perturbations in the skeleton will not change the results. The skeletonization process is not sensitive to blurring [37], since only the distance field values along the spine will change and these are normalized before matching. The addition of small parts to an object will change the full skeleton, but not the central spine. Since the matching process accommodates part matching, adding limbs to the skeleton will not have an effect on the matching process.

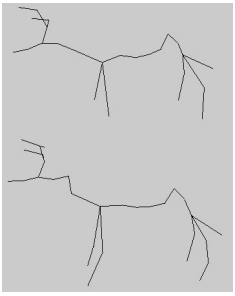

Figure 9. Adding noise: Skeleton of a cow without noise (top image) and with noise (bottom image).

8 Conclusion and Future Work

In this paper, we have demonstrated the feasibility of a 3D shape matching, retrieval and comparison methodology based upon using the "skeleton" of a 3D object and have presented initial results. A skeletal graph is computed directly from the volumetric object and is a 1D approximation of the Medial Surface. We have also presented an improved algorithm for computing a skeletal graph, an enhanced matching algorithm, and a method to visualize the result.

Certainly, the problem of matching is somewhat subjective, and using skeletal graphs is one method to retrieve a possible match. Further testing on a larger database with indexing [31] needs to be done to determine the ideal effectiveness of the matcher. This would be combined with a fully interactive user interface which would allow the user to refine the matching process. Other enhancements include improving the combined visualization, utilizing better registration methodologies, and incorporating an object morpher which after matching would compute the best morph from one object to the other based upon the skeletons. The question of computing the "best" skeletal graph from a 3D object is also still and open area of investigation [9, 1].

In addition, we have been investigating other, more exact, skeletonization algorithms and preliminary results are available on the web site [35]. The skeletons are computed using the potential field [2] (a continuous approach) and are smoother and more robust to noise. Furthermore, the new method does not use the minimum spanning tree so connectivity errors are avoided. In addition, important topological quantities, (such as joints, etc) are identified correctly by the method. This should improve the matching process significantly.

The computation of the skeleton is fast and is proportional to the number of voxels in the model. The benefits of the skeletonization matching are in its ability to do part matching, articulated matching, to compute orientation differences for registration, and the intuitive nature of the skeleton which can be understood by the user. The skeleton is a qualitative tool for matching and analyzing 3D volumetric objects.

9 Acknowledgments

We would like to thank Vikas Singh, Xioasong Yuan, Jeff Dwoskin, Yaakov Kesselman, and Ali Shokofandeh for their help. This work was done in the Vizlab at the CAIP Center, Rutgers University. The laboratory gratefully acknowledges the support from NSF (0082634,0118760).

References

[1] A. Liu and E. Bullitt and S. Pizer. 3D/2D Registration via Skeletal Near Projective Invariance in Tubular Objects. In W. M. Wells, A. Colchester, S. Delp,, editor, *Medical Image Computing and Computer-Assisted Intervention- MICCAI '98. Lecture Notes in Computer Science 1496: Medical Image Computing and Computer-Assisted*, pages 952–963. Springer, New York, 1998.

[2] N. Ahuja and J.-H. Chuang. Shape representation using a generalized potential field model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):169–176, 1997.

[3] R. Bansal, B. Geiger, A. Banihashemi, and A. Krishnan. Integrated Segmentation, Registration and Visualization of Multimodal Medical Image Datasets. In *Work-In-Progress IEEE Visualization 2000*, Oct. 2000.

[4] G. Barequet and M. Sharir. Partial Surface and Volume Matching in Three Dimensions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19, 1997.

[5] P. Besl. A Method for Registration of 3D Shapes. *IEEE Trans. On Pattern Analysis and Machine Intelligence*, 14(2):239–256, February 1992.

[6] J. Bloomenthal and C. Lim. Skeletal methods of shape manipulation. In *Proc. Shape Modeling and Applications*, pages 44–47. IEEE, 1999.

[7] H. Blum. *A Transformation for Extracting New Descriptors of Shape*, pages 362–380. MIT Press, 1967.

[8] G. Borgefors. On Digital Distance Transforms in Three Dimensions. *Computer Vision and Image Understanding*, 64(3):368–376, November 1996.

[9] S. Bouix and K. Siddiqi. Divergence-Based Medial Surfaces. In *ECCV '00*, 2000.

[10] C. Burbeck and S. Pizer. Object Representation by Cores: Identifying and Representing Primitive Spatial Regions. *Vision Research*, 35(13), 1995.

[11] I. Cox, M. Miller, T. Minka, T. Papathomas, and P. Yianilos. The Bayesian Image Retrieval System, Pichunter: Theory, Implementation and Psychophysical Experiments. *IEEE Transaction on Image Processing*, 9(1), Jan. 2000.

[12] F. Dachille and A. Kaufman. Incremental Triangle Voxelization. In *Graphics Interface 2000*, May 2000.

[13] M. Elad, A. Tal, and S. Ar. Content based Retrieval of VRML Objects - An iterative and Interactive Approach. In *The 6th Eurographics workshop in Multimedia*, Manchester UK, September 2001.

[14] N. Gagvani and D. Silver. Parameter Controlled Volume Thinning. *Graphical Models and Image Procesing*, 61(3):149–164, May 1999.

[15] N. Gagvani and D. Silver. Animating Volumetric Models. *Graphical Models*, Mar. 2002.

[16] Google image search engine, http://www.google.com, 2002.

[17] T. He and L. Hong. Reliable Navigation for Virtual Endoscopy. In *Proc. Medical Imaging*. IEEE, October 1999.

[18] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. Kunii. Topology Matching for Fully Automatic Similarity Estimation of 3D Shapes. *ACM SIGGRAPH 2001 Proceedings*, Aug. 2001.

[19] X. Li, T. Woom, T. Tan, and Z. Huang. Decomposing Polygon Meshes for Interactive Applications. In *ACM Symposium on Interactive 3D Graphics*, Mar. 2001.

[20] A. Liu, S. Pizer, D. Eberly, B. Morse, J. Rosenman, E. Chaney, E. Bullitt, and V. Carrasco. Volume Registration Using the 3D Core. *Visualization in Biomedical Computing '94. SPIE 2659*, pages 217–226, 1994.

[21] N. Amenta and S. Choi and R. Kolluri. The Power Crust, Unions of Balls and The Medial Axis Transform. *Int. J. Computational Geometry and Its Applications*, 2002.

[22] W. Niblack, P. Gibbons, and D. Capson. Generating Skeletons and Centerlines from the Distance Transform. *CVGIP : Graphical Models and Image Processing*, 54(5):420–437, September 1992.

[23] F. Nilsson and P.-E. Danielsson. Finding the Minimal Set of Maximum Disks for Binary Objects. *Graphical Models and Image Processing*, 59(1):55–60, January 1997.

[24] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin. Matching 3D Models with Shape Distributions. In *Shape Modeling International*, Genova, Italy, May 2001.

[25] Princeton Shape Retrieval and Analysis, 3D Model Search, http://shape.cs.princeton.edu/search.html, 2002.

[26] A. Rosenfeld and J. Pfaltz. Distance Functions on Digital Pictures. *PR*, 1(1):33–61, 1968.

[27] T. Saito and J. Toriwaki. New algorithms for Euclidean Distance Transformation of an n-Dimensional Digitized Picture with Applications. *Pattern recognition*, 27:1551–1565, 1994.

[28] H. Schirmacher. *Extracting Graphs from Three Dimensional Neuron Datasets*. Diploma thesis, Universität Erlangen-Nürnberg, 1998.

[29] D. Shaked and A. Bruckstein. Pruning Medial Axes. *Computer Vision and Image Understanding*, 69(2):156–169, Feb. 1998.

[30] A. Shokoufandeh and S. Dickinson. Applications of Bipartite Matching to Problems in Object Recognition. In *Proceedings, ICCV Workshop on Graph Algorithms and Computer Vision*, September 1999.

[31] A. Shokoufandeh and S. Dickinson. A Unified Framework for Indexing and Matching Hierarchical Shape Structures. In *Proceedings, 4th International Workshop on Visual Form*, Capri, Italy, May 28–30 2001.

[32] A. Shokoufandeh, S. Dickinson, C. Jonsson, L. Bretzner, and T. Lindeberg. On the Representation and Matching of Qualitative Shape at Multiple Scales. In *Proceedings, European Conference on Computer Vision, Copenhagen,* , May 2002.

[33] A. Shokoufandeh, S. Dickinson, K. Siddiqi, and S. Zucker. Indexing using a spectral encoding of topological structure. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 491–497, Fort Collins, CO, June 1999.

[34] K. Siddiqi, A. Shokoufandeh, S. Dickinson, and S. Zucker. Shock graphs and shape matching. *International Journal of Computer Vision*, 30:1–24, 1999.

[35] Vizlab: http://www.caip.rutgers.edu/vizlab.html.

[36] I. S.Svennson and G. S. di Baja. Curve skeletonization of surface-like objects in 3d images guided by voxel classification. *Pattern Recognition Letters*, 23(12):1419–1426, 2002.

[37] R. Veltkamp and M. Hagedoorn. State-of-the-art in shape matching, Technical Report UU-CS-1999-27, Utrecht University, the Netherlands, 1999.