A robust approach for constructing a graph representation of articulated and tubular-like objects from 3D scattered data

Naoufel Werghi

College of Information Technology
Dubai University College
Dubai, P.O Box 14143, UAE
Email: nwerghi@duc.ac.ae

Abstract

This paper describes an approach for constructing a graph representation of 3D objects and more particularly of articulated and tubular-like objects. For objects without holes, this representation is a tree structure that encodes the object template while being invariant to global and local rigid transformation. The approach described in this paper has some interesting aspects: 1) It operates on raw 3D scattered data points, without any pre-processing stage. 2) It has low computational cost. 3) It is robust against irregular data point distribution and data deficiencies. This graph representation can be used in various applications such as object coding, recognition, and segmentation.

Key words: Graph-based 3D shape representation, articulated and tubular-like objects, Reeb Graph, geodesic distance, graph visualization.

1 Introduction

3D Object shape abstraction and encoding has been receiving an increasing attention in recent years. It is fuelled by the advances in 3D shape imaging technologies and the proliferation of 3D object model databases, where 3D shape representation plays a fundamental role in model retrieval and shape matching. In the literature (Tangelder and Veltkamp 2004) shape representation can be broadly categorized in three categories, namely, feature based representations, graph based representations, and other representations. Feature based representations encompass only pure geometry information of the object. In contrast, graph based representations, which use a graph showing how shape
components are linked together, embed in addition to some geometric information, topological and structural meanings that are quite suitable for high level processing. Graph based representations include three families, namely, model graph, skeletons, and Reeb-Graph. Model graph representations are especially suitable for man-made objects (i.e CAD/CAM models) and are generally difficult to apply for models of natural shape. Skeletons can be applied to wider shapes including animal and human shapes. Skeleton constructions have been approached using the medial axis model (Chuang et al 2000; Naf et al 1996; Siddiqi et al 2002; Bouix et al 2005) and the distance transform (Gavani and Silver 1999; Sanniti di Baja and Svensson 2002); (Svensson and Sanniti di Baja 2002).

Reeb-Graph, introduced by Reeb (Reeb 1946), is a particular skeleton determined using a continuous scalar function defined on an object surface. The main characteristics of a Reeb-Graph are 1) One dimensional graph structure and 2) Invariance to both global and local geometric transformations. These characteristics make it suitable for articulated objects. Reeb-Graph has been used in many applications such as shape coding (Tai et al 1998), shape matching (Hilaga et al 2001), surface compression (Biasotti et al 2002), and human-body scan segmentation (Xiao et al 2003a; Xiao et al 2003b; Xiao et al 2004).

In this paper we propose a method for constructing and visualizing a Reeb-Graph of a 3D object. Compared to previous methods, our method is characterized by the following features: 1) It operates on raw 3D data, i.e., cloud of scattered data points (in contrast to methods that require mesh-model data). 2) It is robust against data deficiencies such as irregular distribution reflected by gaps and holes. 3) It has low computational costs. The approach targets objects having tubular-like shapes or a blending of generalized cylinder shapes and assumed the object surface topologically continuous.

The rest of the paper is organized as follows: Section 2 gives an overview of the approach. Sections 3, 4, 5, and 6 describe the different stages of the approach. Experimental results are discussed in section 7. Finally, in section 8, conclusions are drawn and further research work is suggested.

2 Overview of the approach

The approach operates on a set of 3D scattered data points representing the object shape. It involves four main stages. These are depicted in Figure 1:

1. Computation of the level-sets: In this step, a scalar function map is computed over the set of the data points of the object surface (b) and level-sets representing isovalued points (points having the same scalar function value) are extracted (c).
2. **Construction of a connectivity graph:** Here, each level-set is decomposed into subsets of connected data points according to a given connectivity criteria. We call these subsets level-set curves. The level-set curves are then mapped into a connectivity graph (d), where a node represents a level-set curve and an edge represents a connection between two adjacent level-set curves, i.e., level-set curves belonging to two adjacent level-sets. The nodes in that graph are arranged by ascending horizontal levels which represent the values of the scalar function at the level-sets determined in stage 1. Thus, nodes corresponding to level-set curves which are part of the same level-set will be placed at a same level.

3. **Extraction of joints and branches:** In this stage, the connectivity graph is analyzed to locate the joint nodes and to segment the connectivity graph into groups that correspond to the object branches (e). This stage outputs a particular Reeb-Graph, namely, a tree structure (f) in which nodes represent the branches and edges represent the joint nodes. The tree structure reflects the assumption that the object does not represent holes.

4. **Visualization:** The Reeb-graph of the object is automatically visualized (g) using the tree-structure obtained in the previous stage. This graph encodes the object branches and parts while describing the evolution of the scalar function across them.

![Fig. 1](image)

Fig. 1. (a): A 3D object. (b) grey level mapping of the scalar function on the object surface. (c): level-sets of the scalar function. (d): The connectivity graph. (e): Determination of the joint nodes and the branches. (f): The tree structure. (g): Visualization of the Reeb-Graph.
3 Computation of the level-sets

Given a set of data points $V$ and a scalar function: $F : X \mapsto \mathcal{R}$ where $X \in \mathcal{R}^3$ is a data point, level-sets in discrete space are formally defined by $\{X \in V, F(X) = C_k\}$ where $C_k, k = 1 : m$ is a set of discrete values ranging from the minimum value to the maximum value of the function $F$ in the domain $V$. To ensure a stable representation, the scalar function should be invariant with respect to rigid transformations. The curvature function satisfies this condition, however, it is highly sensitive to noise and data corruption. A function employing the geodesic distance (Mitchell et al 1985) is more appropriate as such distance is quite resistant to data corruption in addition to being invariant to rigid transformations. We utilize the function defined by $F(X) = gd(X, S)$ which returns the geodesic distance from a point $X$ to a source point $S$. In the literature, the Dijkstra algorithm (Cormen et al 1990) has been the most popular tool for computing geodesic distances between a group of points and source point. However it has high computational cost. So we rather developed an efficient algorithm, tailored for our application. The algorithm deploys a wavefront propagation technique, which is based on the following principle: Given a centred wave on a manifold, all the points on the wavefront have the same geodesic distance to the wave centre and form thus a level-set. Our wavefront propagation algorithm operates on a binary voxel grid since it is easy to define a neighborhood in voxel space and to traverse connected voxels. Due to these well-behaved properties, wavefront-propagation on a voxel grid can have a very simple mathematical form as follows:

$$
\begin{align*}
W_0 &= \{v_s\} \\
W_{i+1} &= (W_i \oplus e - (W_i \oplus e) \cap S_i) \cup \overline{S_i}
\end{align*}
$$

where: $W_i$ is the wavefront generated on the $i^{th}$ iteration of the algorithm; $v_s$ is the source voxel; $S_i$ is the set of all 1-valued voxels visited at the iteration $i$ and located at the same geodesic distance from the source voxel $v_s$. $\overline{S_i}$ is the complement set of $S_i$. $\oplus$ denotes the morphological dilation operator and $e$ is a $3 \times 3 \times 3$ structuring element composed of 27 1-valued voxels. At the beginning, the wavefront is the source voxel associated to the source point, then the wavefront iteratively spreads on the voxelised surface. In each iteration, the wavefront is the level-set containing voxels with the same geodesic distance. The attractive aspect of this technique is that it simultaneously extracts the level-sets while computing the scalar function $F$. It is easy to prove that the computational complexity in each iteration is $O(n_i)$, where $n_i$ is the number of voxels in $W_i$. Therefore the complexity of the whole algorithm is $O(N)$, where $N$ is the number of all 1-valued voxels.
The source point $S$ can be selected manually or determined automatically following these steps 1) Choose a source point at random from the set of data points 2) compute the geodesic distance map 3) Choose the point corresponding to the maximum geodesic distance value. For articulated and tubular-structured objects, such point will be located at the extremity of the object parts. This presents the advantage of maximizing the range of the geodesic distance.

4 Construction of a connectivity graph

For a perfect data, a level-set would be a compact set of connected points. For real data characterized by a non-uniform distribution and gaps, the level-set is rather fragmented into sets of connected points. These sets, which we call level-set curves, are conceptualized by the following definitions.

**Definition 1 (connectivity of point sets):** Two point sets $P = \{p_i\}, \ i = 1..m$ and $Q = \{q_j\}, \ j = 1..n$ are defined as connected if $\exists p_i \in P$ and $\exists q_j \in Q$ such that $|p_i - q_j| \leq \tau$.

Where $|p_i - q_j|$ denotes the distance between points $p_i$ and $q_j$ and $\tau$ is a given threshold. The above definition also holds for the connectedness between two points for the particular case where the sets $P$ and $Q$ contain a single point each.

**Definition 2 (connective point set):** A point set $C$ is connective if $\forall$ subset $\Omega \subset C$ and $\Omega \neq \emptyset$, $\Omega$ and $\bar{\Omega}$ are connected. Here $\bar{\Omega}$ denotes the complement of $\Omega$ in $C$. Definition 2 defines a 'tight' point set in which all the points are connected.

**Definition 3 Level-set curve:** A level-set curve is an isovalued connective point set, that is a group of points, that have the same scalar function value, and which forms a connective point set.

At the implementation level, the threshold $\tau$ used in definition 1 is set to the resolution of the 3D data points. The resolution is estimated as follows: we determine the distribution of the distance values of the most closed pairs (i.e. nearest neighbor tuples) of data points over a large set of data. This will permit to construct the 3-D density histogram. The median value or the weighted average are reasonable estimates of the resolution, more elaborated techniques can be used however (Scott and Sain 2004).

The connectivity graph is an oriented graph where a node represents a level-set curve and where an edge represents a connection between two adjacent level-set curves, (i.e. two level-set curves belonging to two adjacent level-sets).
The connection between the two level-set curves is established according to definition 1. Ideally, when we assume a clean data, the connectivity graph will be reduced to a tree structure. But practically, the deficiencies of the data produce topological disturbance that causes false joint nodes. This results in a corrupted graph structure. Figure 2 shows an example illustrating this aspect in more details. For simplicity, the example represents a cylinder-like shape and covers only a portion of three adjacent level-sets. If we assume an ideal data, each level-set consists of an organized set of connected points Figure 2(a). So each one will represent a single level-set curve. This will result in a graph composed of three nodes (Figure 2(b)). For real data, a level-set might be composed of several level-set curves because of data corruption, leading to several nodes per level-set. The example in Figure 2(c) shows three level-set curves for level-set 3 and two level-set curves for each of level-set 1 and level-set 2. The three initial nodes $L_1, L_2, L_3$ have degenerated into seven nodes $l_1, l_2, ..., l_7$. Setting afterwards the connections among the nodes (leads to a disorganized graph (Figure 2(d)). Therefore, at the scale of the object the connectivity degenerates into chaotic graph representing false joint nodes and false branches. The challenge therefore, is to be able to determine the correct joints and branches from such degenerated graph to obtain a representation faithful to the topological structure of the object. This will be described in the next section.

5 Extraction of joint nodes and branches

The strategy adopted in this stage is based on the following analysis: In the connectivity graph we identified three primary topological patterns. These
patterns are called O-type, Λ-type and Y-type. For example, The group of nodes \((l_7, l_4, l_2, l_1)\), \((l_6, l_4, l_1, l_3)\), and \((l_6, l_5, l_3, l_1)\) represent a Λ-type, an O-type, and a Y-type respectively. O-type comprises two joint nodes connected by two branches. This pattern reflects data corruption (gaps, missing data) because we assumed that the object does not contain holes and therefore this pattern is simply ignored. The Λ-type and Y-type are topologically identical as each represents three branches that meet at the joint node. We can reduce the number of patterns to a single one by considering a topographic constraint when constructing the connectivity graph. Indeed, by arranging the nodes in ascending levels, where these levels represent the values of the geodesic function at the different level-sets, and by placing at each level the nodes associated to level-set curves which belong to a same level-set, only Y-type patterns can figure in the connectivity graph. To distinguish genuine Y-types from false ones which are inferred by data deficiencies, we impose the length of a branch (evaluated in terms of inferred number of levels) associated to a true Y-type to be above a minimum value.

Based on these considerations, the following criteria are used to identify a ‘true’ Y-type which consists of ‘true’ branches and a joint node: 1) A ‘true’ branch is upward and must satisfy: \(|L_{\text{max}}(\text{branch}) - L_{\text{min}}(\text{branch})| > L_{\text{lim}}\) where \(L_{\text{max}}/L_{\text{min}}(\text{branch})\) denotes the maximum/minimum levels inferred by the branch in the connectivity graph. \(L_{\text{lim}}\) is a threshold that represents the minimum length allowed for a branch. 2) A true joint node has at least two branches.

The setting of \(L_{\text{lim}}\) value is very crucial. In effect a low value might generate false branches whereas a high value will causes actual branches to be lost. Here we note that a genuine choice repose values of \(L_{\text{lim}}\) should stand on a prior knowledge of the sizes of the segments that form the object relatively its whole size. For example when dealing with animal shapes, we know that the minimum size of a functional part cannot be less one tenth of the whole size. A reasonable value of \(L_{\text{lim}}\) can then be set based on that proportion.

Based on the details described above, the detection of the true joint nodes is accomplished using the algorithm given below. This algorithm browses the connectivity graph level by level starting from the highest level that corresponds to the level-set associated with the maximum value of the geodesic distance function. In a second phase the true branches are extracted based on the locations of extreme nodes and joint nodes. Having located the joint nodes and branches, a tree structure is constructed afterwards where nodes represent branches and edges represent joint nodes linking the branches.

**Notation:**

*Branch*: A group of connected nodes.

*Branch(Node)*: The Class containing the Node.
For each level in the connectivity graph
  For each node at that level
    Extract from the upper level the group of nodes \((l_1, l_2, \ldots, l_m)\) connected to that node
    If \(m = 0\)
      Add this node to the list of extreme nodes
      \(\text{Branch}(\text{node}) := \{\text{this node}\}\)
    Else
      From the set: \(\{\text{Branch}(l_i), i = 1..m\}\) select the Branches verifying:
      \(\{ B_j : \left| L_{\text{max}}(\text{Branch}(l_j)) - L_{\text{min}}(\text{Branch}(l_j)) \right| > L_{\text{lim}}, j = 1..n \}\)
      If \(n \geq 2\)
        add the nodes \(l_j, j = 1..n\) to the list of Joint nodes
      End If
    \(\text{Branch}(\text{node}) = \{\text{Branch}(\text{node})\} \cup \{\text{Branch}(l_1)\} \cup \ldots \cup \{\text{Branch}(l_m)\}\)\)
  End If
End for
End for

6 Visualization

In this stage, the topological structure embedded in the Reeb-Graph of the object is visualized. The tree structure outputted by the previous stage is browsed in a depth-first fashion. At each visited node, the associated branch is mapped into a 2D curve where the x-coordinate and the y-coordinate represent a level-set curve and its corresponding level in the connectivity graph. We have to mention here that the orientation of the 2D curve only reflects the evolution of the geodesic distance at the associate branch and does not contain geometrical information as it is the case in skeletons.

7 Experiments

We applied our approach to a variety of objects acquired from different sources. Figure 3 shows results obtained with animal shapes. These models were acquired from Princeton Benchmark\(^1\). We can observe that the resulting graphs reflect correctly the topology of the models. The graphs of the dog and the camel present each a main branch and five ramifications that correspond to the limbs and the tail. The graph of the horse shows only four ramifications as the corresponding model does not include a tail. One might ask if using a

\(^1\) http://shape.cs.princeton.edu
smaller values of $L_{lim}$ (Section 5) would permit to detect finer details of the shape (e.g. the ears of the dog or the horse). This is possible under ideal conditions (i.e. dense and clean data), however practically because of data irregularities, lowering the $L_{lim}$ might cause undesired noise branches in the graph. To ease these effects, one solution would be to clean-up the data and increase the resolution in a pre-processing stage.

Figure 4 shows three models of blood vessels, composed of three, five and 8 branches respectively. These models were obtained from the CVMT Lab\textsuperscript{2}. The second row in Figure 4 depicts the corresponding graphs. We can see that all the trees exhibit a correct representation in terms of structure and number of branches. We conducted other trials on the third blood vessel model to assess the stability of the representation with respect to rigid transformations and change of source point location. The results are depicted in Figure 5. In the first trial(a) we applied random rotation of the model and kept the same source point. The corresponding tree remains unchanged. This illustrates the invariance of the geodesic distance function to rigid transformations. In the two other trials (b and c) we rotated the model and changed the location of the source points. While the resulting trees look having different configuration, they do preserve the same number of branches and nodes, reflecting thus the stability of the representation with respect to the topological structure of the model. In a second series of experiments we tested the robustness of the approach with respect to data deficiencies. It is worth to mention that the

\textsuperscript{2} Computer Vision and Media Technology Lab, Aalborh University, Denmark. www.cvmt.dk
Fig. 4. Instances of blood vessels and their graph representations.

Fig. 5. Grey level mappings of the geodesic function corresponding to different source points (marked by a "+") for a rotated blood vessel model and the resulting graph representation.
data used in the experiments are characterized by irregular distribution as shown by the zoomed area of the camel in Figure 7. Firstly we corrupted some objects by a boundary noise as depicted in the first row of Figure 6. The second row shows the corresponding trees. We can observe that the trees keep a correct structure. To further check the robustness of our method we intentionally corrupted some models by creating artificial holes at different locations of their surfaces (Figure 7). Despite these severe alterations, the approach produced correct graphs as depicted by the figure. These results illustrate particularly that the connectivity graph analysis described in Section 5 succeeds in rejecting the O-type nodes in the connectivity graph.

Other experiments have been conducted on both synthetic and real articulated objects to test how the approach cope with deformation altering the shape of an object. Figure 8: first row, shows instances (the first four objects) of a synthetic object having undergone various deformation. The next row depicts the
resulting trees (the first four trees). We can see that all the trees have the same structure. Some trees show different orientations for some branches, but this does not reflect any geometric properties as the branch orientation is set by the graph visualisation algorithm and do not infer any geometric information.

The third row in Figure 8 shows animated instances of a frog. These instances were generated using the animation software Poser\(^3\). The next rows depict the corresponding trees. The stability of the tree representation is clearly noticed. Figure 9 illustrates instance of shapes that our method cannot handle. The object presents cavities, which infer topological discontinuities. Such discontinuities are not detected by our algorithm as it is assumes continuous surfaces. We plan to address this issue in future work.

The last bunch of tests has been conducted to illustrate the utility of our ap-
proach in a particular applications namely object segmentation. We developed a simple segmentation method based on that graph representation. Basically the method consists in mapping the branches of the tree with the model data. The experiments were conducted with a variety of synthetic and real objects. Results are depicted in Figure 10. We can see that the different segments in each object have been retrieved, not accurately however as it can be seen, for instance in the first synthetic object (H-shape) in row 1, and also in the dog and the horse. We believe a more thorough segmentation approach involving analysis of the level-sets around the joint nodes of the graphs would yield better segmentation.

The approach has been implemented with Matlab on a 1.2 Ghz Pentium III machine. The code, however, is not optimized. To give an idea about the running time, the graph construction of the the third blood vessel model (which contains 14 960 points) took about 8 seconds.

8 Discussion and Conclusion

In this paper we proposed an approach for automatically constructing a topological representation of 3D objects The main features of this approach are: 1) It operates on crude 3D scattered data. 2) It is robust against irregular data point distributions and severe data deficiencies such as gaps. 3) It involves an efficient technique that computes simultaneously the geodesic function and the associated level-sets. This technique demonstrates a novel algorithm characterized by a low computational costs. The experiments conducted on a variety
of objects and shapes confirmed the effectiveness and the robustness of the approach. We illustrate the applicability of the proposed graph representation in object segmentation. Yet it can be also exploited for data registration and object recognition.

Compared to Reeb-graph based method are approach is more efficient thanks to the fast technique used for computing the level-sets. It is also more robust as it can cope with severe data alteration as it has been illustrated in the experiments. To the best of our knowledge we are not aware of any other Reeb-Graph construction technique that can handle effectively such altered data.

With respect to skeleton construction, our approach is not qualified to compete with other approaches (e.g. the medial-axis technique), as it is essentially a graph construction approach intended to deliver a representation that encompasses only topological and a structural information. However in the medial-axis construction approaches reported in the literature, it is not clear how and to what extent they can cope with instances of data deficiencies as the ones we did test in our experiments (Figure). Our assumption is that unless detected and handled properly such gaps will cause distorted constructed axes that might result on uncorrect skeleton.

Our approach presents some limitation however. It needs some user intervention to determine the source point and to set adequate value of $L_{lim}$ (Section 5). So far the approach cannot handle objects presenting topological continuities (e.g. cavities). We plan to address these issues in the future. For the scalar function in particular, we need to investigate source point-invariant functions. This work can be also further explored in other directions. We plan to investigate the extension of our approach to skeleton extraction. Such skeleton extraction will inherit the robustness of our approach against severe data deficiencies.

References


