Fast Point Cloud Skeleton Extraction via Octree-graph

Jia Cao[1]
School of Computer and Information Technology, Beijing Jiaotong University ,Beijing, 100044, China
E-mail: `boafly@outlook.com`

Hongli Xu
School of Computer and Information Technology, Beijing Jiaotong University ,Beijing, 100044, China

We present a fast algorithm to implement curve-skeleton extraction from imperfect point sets. The algorithm works directly on the point clouds data without transforming the point cloud object into a volumetric or mesh form, which can be readily extracted qualified skeletons by plenty of mature methods. Specially, we first extract a novel graph called octree-graph from the original point cloud data, then reduce it into a linear skeleton, and simultaneously embed it into the original point cloud. The final skeleton can possess excellent centeredness and topological correctness on a variety of three-dimensional models.

CENet2015
12-13 September 2015
Shanghai, China

[1]Speaker

1. Introduction

A point cloud skeleton refers to a “compact” 1D structure that stands for a reduced version of the geometry and topology of a 3D object. Applications which utilize the curve skeletons include virtual colonoscopies and virtual endoscopies[1], segmentation , shape matching and retrieval , collision detection , surface reconstruction , computer animation , 3D object registration and visualization , therefore, the extraction of skeletons plays a very important role in the field of computer graphics and visualization[2-6].

There has been much work on extracting skeleton from 2D shapes and 3D models, operating exclusively on shapes specified by closed polygonal meshes or complete volumetric points[7-9]; however, object 3D representations may be firstly acquired by modern measurement instruments, such as laser scanning, in the form of point clouds. Thus this form of data has posed a few inevitable algorithmic challenges because of noise, under-sampling, varying point density and huge amount of data. Under these circumstances, computing a skeleton from imperfect point clouds data is fascinating; therefore, we propose a robust algorithm to extract the object skeleton, here called Octree-graph(OG) skeleton, directly on unorganized point sets. The algorithm incorporates three main procedures. Firstly, an OG, delegating the connectedness between the sub-cells, is extracted from the octree which has divided the original point cloud data into sub-cells. In order to preserve original objects’ topology, a robust criterion is used to decide proper connectedness in the OG. In the second step, the OG is reduced to a 1D structure called OG skeleton. The third step of strategy, simultaneous performance with the second step, is to embed the OG graph into the original point sets. The centeredness is improved by an existing embedding strategy with respect to the original object [11]. The contribution of this algorithm contains a new OG graph construction and a new graph reduction method that automatically incorporates the approximate local structure skeleton by using suitable methods. These approximate skeletons could generate the final skeleton by using a kind of embedding method.

2. Octree-graph Skeletonization Algorithm

2.1 Extraction of Octree-graph

Octree Generation. We use a self-adaptive octree space subdivision algorithm to split the point cloud data into smaller sub-cells. Each subdivision process will stop when the number of point clouds in the sub-cell is less than self-adaptive parameter K, which represents possible maximum points within a sub-cell. Let N represent the total number of the point clouds. As a result, the time complexity of self-adaptive octree subdivision is approximately $O(N^2/K)$, the time complexity of octree-graph construction after subdivision is about $O(KN)$. To minimize overall consumed time, we define a time Eq.1 f(x)

$$f(x)=N^2/K+K\cdot N+a \tag{2.1}$$

We get $K=\sqrt{N}$ by deducing extremum from Formula (2.1).

Extraction and labeling of Octree graph. Then we construct octree graph from the above-mentioned octree subduction point cloud. The output will afterwards be reduced to the final OG skeleton. In this paper, we use center of the whole points belonging to a cell to represent the vertex and then connect two vertices in adjacent cells by an edge if they satisfy the connection criteria. This criterion is shown in the Fig.1.

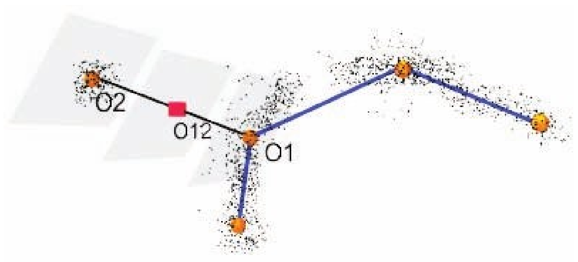


Figure 1: Octree-graph Vertices Connection Criterion

O_1 and O_2 are the central points of two adjacent cells Φ_1 and Φ_2 . O_{12} is the central point of line segment O_1O_2 . Three selected planes T_1, T_2 and T_{12} , respectively passing through points O_1 , O_2 and O_{12} , are vertical with the line segment O_1O_2 , and d_1 , d_2 and d_{12} represent the average value of the squared distances from all the points in Φ_1 , Φ_2 and $\Phi_1 \cup \Phi_2$ to planes T_1, T_2 and T_{12} . Under ideal conditions, in order to indicate a proper connection between two corresponding cells, the $\sqrt{d_{12}}$ would be at least $\frac{1}{4}$ of the distance between O_1 and O_2 . In this sense, the paper uses the following criterion as show in Eq.2.2:

$$\frac{1}{16} d_{12} \leqslant min(d_1, d_2) \tag{2.2}$$

Now that we have defined and extracted the corresponding OG, then we must consider how to define each edge's direction. Each edge of the OG uses its corresponding direction vector as the direction label.

The resulting OG should be a bi-dirtional graph and each edge has two direction labels for different view points of the two incident vertices. For example, there are two points $O_1(x_1, y_1, z_1)$ and $O_2(x_2, y_2, z_2)$, where $x_1 < x_2$, $y_1 = y_2$ and $z_1 = z_2$; the label of edge e_{12} is (1,0,0) , while that of e_{21} is (-1,0,0).

2.2 Graph Reduction

This step merges some adjacent vertices which meet with specified criteria into a single vertex. The implementastions on the graph are intuitively regarded as the collapse of two edges incident to the same vertex. Octree graph's local contraction is completed by operating local structural units. In order to better quantify the process, we define vdim as the number of different labeled edges connected to a vertex, vdr the sum of direction vectors corresponding to edges with different labels and operator $\oplus$ the implementation emerging two points. Specially, given $vdr(v) = (x_1, x_2, x_3)$, if Cartesian entries are not zero for any associated edge label, but the value of x_i in v is always zero, then it's called as the primary direction. On the basis of the said set of definitions, we can validly merge vertices. Every vertex is inevitably adjacent to more than one other vertex. In order to ensure the primary direction of the merged vertex unchanged, we can know which vertex is to be merged by it first.

Besides, various local structure units are connected together with two configurations, which are defined as E-Connection and V-Connection in this paper. The structures of these two kinds of configuration are defined respectively.

E-Connection. Given that u_i and u_j are two adjacent vertices, and $vdim(u_i) \leqslant vdim(u_j)$, if u_i and u_j satisfy the following conditions:

(1) $vdim(u_i \oplus u_j) \leqslant max(vdim(u_i), vdim(u_j))$;

(2) $vdr(u_i) \neq (0,0,0)$, u_i and u_j are connected at the non-primary direction of u_i ;

V-Connection. If two vertices u_i and u_j are both adjacent to a third vertex u_c , and they satisfy the following conditions:

(1) u_iu_c and u_ju_c have exactly the same direction markers;

(2) $vdim(u_i \oplus u_j) \leqslant max(vdim(u_i), vdim(u_j))$.

An E-Connection can be reduced to a V-Connection and a 1D skeleton can be extracted from a V-Connection structure; therefore, the reducing process cannot stop until the final OG skeleton can be received.

2.3 Graph Embedding

In order to keep the final skeleton's centrality, we utilize an embedding method described in Document to make the skeleton embed in the original point cloud when it is in the calculating process of OG skeleton [11]. Every vertex in octree-graph has an initial weight of w, which represents the number of points in the vertex's corresponding subspace. In the process of merging, we use weights of vertices v_1 and v_2 and their dimensional space coordinates to calculate the weights and coordinates of the new emerged vertex v_{new}, so that this process can be iterated on until the convergence to linear skeleton. The weights and its coordinates of new emerged vertex v_{new}'s are shown in Eq.3:

$$w_{new} = w_1 + w_2 \quad , \quad v_{new} = \frac{w_1 \cdot v_1 + w_2 \cdot v_2}{w_1 + w_2} \tag{3}$$

3. Results and Discussions

In this section, we use our algorithm for the experiments with a variety of three-dimensional model of Princeton University's three-dimensional model database and discuss some of its properties We then compare the time cost of our algorithm with previous methods.

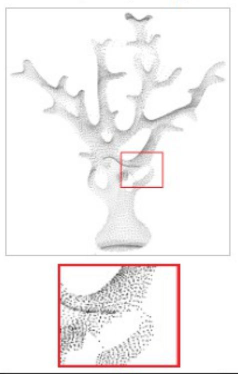
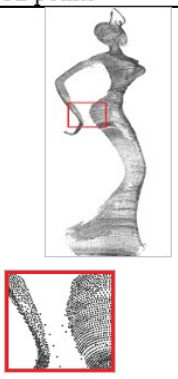
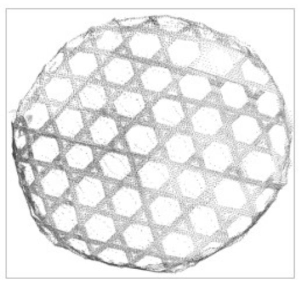
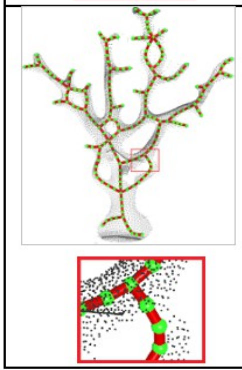
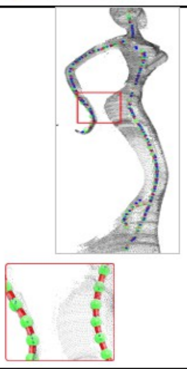
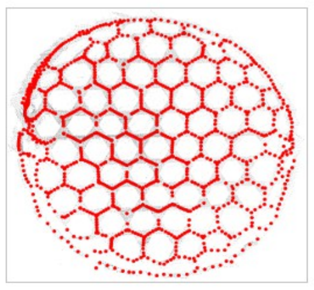
Coral/1,000,079 points	Lady/49,832 points	Basket/45,932 points
		
		

Figure 2: 3 Test Objects. First coral with lack of points and second with mixed structure which contains much noisy and third with high-genus structure

We test our algorithm on different geometric and topological characteristics objects such as high-genus and heavy branching structure; moreover, all the input data to our algorithm are unorganized and un-oriented raw scans without preprocessing for noise or missing data. This input indicates various extent of datasets including much noise, non-uniform structure or missing data.

As we all see from Fig.2, the result indicates that our algorithm is capable of handling so varying kinds of objects. In particular, the first object in Fig.2 demonstrates the good robustness to original points cloud with missing data. Besides, we also verify the robustness of our algorithm to a mixed structure which contains majority of noisy in the second object in Fig2. As you see in the circumstances between the lady's hand and hip containing a lot of noise, the skeleton extracted by using this paper's method is able to correctly reflect the topology structure of the original point cloud, and vividly show the saddle-points as well as extreme points of the object. Eventually, we justify the handing of high-genus structure object and it can yield a satisfactory skeleton.

As we have seen from above discussion, one of the great improvements of our algorithm is that it can directly handle raw scans data without preprocessing methods such as denoising and mesh simplification, which are the necessary procedures as to traditional methods. As for this, our research about skeleton extraction is meaningful and makes some c practical contributions.

Comparison to previous classical methods in cost time: we have proved that our algorithm features more robustness and adaptability than the traditional algorithm, then we will justify that it is also more efficient.

We have selected some classical objects to record the cost time of previous methods and mine. Table 1 shows the running time of some skeleton extraction examples. All data are recorded on an Intel Core i5-3470 machine with 4GB memory.

Model	Vertices	Thinning	Distance Field	Reeb Graph	Our Method
Wolf	4344	10	8	13	2
Raptor	25000	61	53	78	10
Neptune	112220	720	458	879	138
Dragon	230000	1689	982	1987	241
Fertility	341607	2389	1345	2768	378
Coral	1000079	8765	4580	11234	1204

Table 1: Running Time (in Seconds) of Some Skeleton Extraction

The presented result indicates that this method is more efficient than previous classical ones.

As a result, this paper's algorithm can produce linear skeletons with sound clarity, connectivity, topological consistency and geometry. In addition, our algorithm can directly extract skeletons on original point cloud data, untreated and accompanied with a lot of noise; besides, it also has efficient running time. These features make it more valuable for practical engineering applications, such as 3D printing, behavior recognition, computer animation and other fields.

4. Conclusion

In this paper, we describe a novel approach for extracting curve skeletons from unorganized, incomplete and un-oriented three dimensional point cloud objects. The algorithm firstly uses suitable octree to subdivide the original point cloud data, then extracts the octree-graph from it, and then uses certain methods continuously to reduce this figure until the desired linear skeleton is eventually produced. Final results show that this method is capable of maintaining good robustness and efficiency for the skeleton extracted from incomplete point cloud data. Our contributions focus on the fact that our algorithm can directly deal with raw point clouds without preprocessing, which is a necessary procedure of previous methods and can improve the speed of handling time.

As to future works, we would like to enlarge our algorithm to analyze imperfect (4D) scans. It is also interesting to concentrate on practical application like animation and deformation of objects.

References

[1] L. Hong,, S. Muraki, A. Kaufman, D. Bartz,, & T. He, *Virtual voyage: Interactive navigation in the human colon*. In Proceedings of the 24th annual conference on Computer graphics and interactive techniques . ACM Press/Addison-Wesley Publishing Co. 27-34(1997).

[2] Li, X., Woon, T. W., Tan, T. S., & Huang, Z. *Decomposing polygon meshes for interactive applications*. In Proceedings of the 2001 symposium on Interactive 3D graphics . ACM, 35-42(2001)

[3] H. Sundar, D. Silver, N. Gagvani, and S. Dickinson, *Skeleton based shape matching and retrieval* . Shape Modeling International, IEEE. 130-139(2003).

[4] F. F. Leymarie, *Three-dimensional shape representation via shock flows*, in BROWN UNIVERSITY, 2003.

[5] N. Gagvani and D. Silver, P*arameter-controlled volume thinning*, Graphical Models and Image Processing, 61(3):149-164(1999).

[6] S. M. Pizer, D. S. Fritsch, P. A. Yushkevich, V. E. Johnson, and E. L. Chaney, *Segmentation, registration, and measurement of shape variation via image object shape*, Medical Imaging, IEEE Transactions on,18 (10): 851-865(1999).

[7] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii, *Topology matching for fully automatic similarity estimation of 3D shapes*, in Proceedings of the 28th annual conference on Computer graphics and interactive techniques, ACM, 203-212(2001).

[8] A. Bucksch and R. Lindenbergh, *CAMPINO — A skeletonization method for point cloud processing,* ISPRS Journal of Photogrammetry and Remote Sensing, 63(1): 115-127 (2008).

[9] Y.-S. Wang and T.-Y. Lee, *Curve-skeleton extraction using iterative least squares optimization*, Visualization and Computer Graphics, IEEE Transactions on,14(4): 926-936(2008).

[10] A. Bucksch and S. Fleck, *Automated detection of branch dimensions in woody skeletons of fruit tree canopies*, Photogrammetric engineering & remote sensing, 77(3): 229-240(2011).

[11] V. Pascucci, G. Scorzelli, P.-T. Bremer, and A. Mascarenhas, *Robust on-line computation of Reeb graphs: simplicity and speed*, in ACM Transactions on Graphics (TOG), 26(3): 58-59(2007).