PROCEEDINGS OF SCIENCE

A Multi-level Method for Fast Computing of All-Pairs 3D Inner-Distance

Wenxiang Chen[1]
Shanghai Jiao Tong University, Shanghai, 200240,China,
Email: chenwenxiang1023@126.com

Jingqi Yan
Shanghai Jiao Tong University, Shanghai, 200240,China
Email: jqyan@sjtu.edu.cn

3D Inner-distance, which is defined as the length of the shortest path between points within 3D shape, is robust to articulation, especially in non-rigid 3D shapes; thus it can be used to construct the robust 3D shape feature descriptor. With the increasing data scale, the time consumption has become a problem in extracting the 3D inner-distances. In order to accelerate the process of extracting the 3D inner-distances, we propose a new method by using a hierarchical graph to compute the inner-distances. In this paper, we first put forward an adaptive-voxelization method based on Octree structure to construct a hierarchical graph; then the multi-level process of extracting 3D inner-distance is thus proposed.

CENet2015
12-13 September 2015
Shanghai, China

[1]Speaker

PoS(CENet2015)039

1. Introduction

With the development of the computer science and technology, 3D shape models have been widely applied in many fields, such as 3D computer games, 3D printing, VR technology and medical-imaging devices, etc.. The feature descriptor of the 3D shapes plays a central role in computer graphics, computer vision and pattern recognition. The shape descriptor most widely applied is based on the distance between point pairs on shape surfaces, like Euclidean distance (ED) or geodesic distance (GD); however, both of them can not reflect the shape articulation well. In this paper, we focus on the inner-distance which is robust to articulation[1].

Moreover, with more and more attention attracted by the Big Data, we initiate to pay attention to the inner-distances of all point pairs, not just sample points; however with the growth of the data scale, the time consumption has become a prior factor when selecting the computing methods. Thus, in this paper, we propose a novel computing method to accelerate the computing process.

Different from the methods as proposed by YS. Liu et al [2], or Chen and Yan [3], which computed the shortest path by only using one graph, our method is based on a hierarchical graph to separate the shape points to different classes according to their space location. The hierarchical graph consists of graphs at different levels from coarse to fine. When extracting the shortest path of two points, according to their space distribution, we choose the corresponding graph and thus accelerate the computation greatly.

In order to construct the hierarchical mapping relation, the first step we should complete is voxelization, which is a process that we transfer shape models from mesh representations to volume ones and through which we can formulate a set of voxels not only containing the surface information of shapes but also describing the attributes within shapes. There are many voxelization methods, such as the one proposed by Wu et al [4], or Mu et al [5]. Both of their methods were based on fixed resolution. Since a lot of the inner voxels share the same attribute, the storage of the voxels is not quite efficient. Chen and Yan [3] proposed an adaptive-voxelization method to resolve this problem; however, the result cannot reflect the hierarchical relation of the voxels. Thus it is not convenient to construct the hierarchical graph we need; consequently, we hereby propose a novel adaptive-voxelization method based on Octree structure.

The remainder of this paper is organized as follows: in Section 2 we describe the adaptive-voxelization in detail; in Section 3 we introduce the multi-level process of extracting the inner-distance; in Section 4, we present the experimental results; and finally in Section 5, we conclude this paper.

2. The Adaptive-Voxelization

2.1 Octree Construction

Our adaptive voxelization method is based on an Octree structure. Octree is a basic data structure usually used to present the 3D space relationship. For each shape model. We construct an axis-aligned octree. We begin with the min bounding box containing the whole shape, then subdivide it into 8 equal-size children. Each octree is split into 8 children recursively when necessary.

Adaptive-voxelization is a process of octree subdividing. We divide the octree node into 8 children recursively when it contains sample points. If the size of the non-empty cells (contain sample points) less than a threshold τ (usually decided by the size of the bounding box and the average triangle size of the mesh model), stop subdividing. The empty cells (contains no sample point) are classified as being inside or outside, which we will present in next chapter.

2.2 Inside-Outside Test

After constructing the octree, the empty octree cells are classified as inside voxel or outside voxel. Including the surface voxel (defined as the voxels at the highest resolution level

which contain the sample points), there are three types of voxel cells in total, namely, the surface voxels, the inside voxels and the outside voxels.

The inside-outside test is mainly used to classify the empty cells. Inspired from the main idea of the inside-outside test proposed by Adams and Dutre [6], we classify the empty cells according to its neighbours.

Our judging method is different from theirs. As we have learned for computer graphics, if the triangle of the surface is pointing to the empty cell, the cell must be outside the shape or it must be inside the shape. In this sense, let c_o represent the centre of the target cell, c_t represent the centre of its closest triangle, and let be n_t the normal of the closest triangle corresponding to its principal neighbour. If $(c_t - c_o) \cdot n_t > 0$, the target cell will be classified as inside voxel; otherwise, it will be classified as the outside voxel.

Moreover, in order to deal with the boundary case, for example, if there is only one non-empty children in the root octree, we can only classify its 3 principal neighbours with the attribute of the rest neighbours we cannot judge because their neighbours haven't been classified. Thus we make some rectification to their testing process. For each root octree, we proceed the test for three times (if all the empty children are classified, break this process); if the three principal neighbours haven't been classified, just pass the target children; afterwards, we finish classifying its neighbours and then determine its attribute. After the two steps, the adaptive-voxelization is complete with some instance shown in Fig. 1.

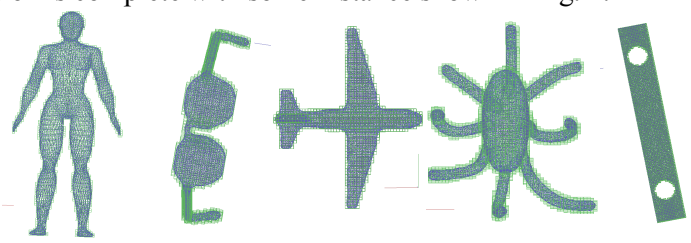

Figure 1 : Some voxelization results: the green cubes represent the voxels and the blue triangles represent the shape meshes.

3. Computing Inner-distances

After finishing the adaptive-voxelization, we transfer 3D shapes from mesh models to volume model. The computing method proposed by Chen and Yan [3] constructed only one graph using all the voxels, then extracted the shortest path between points pairs (voxels pairs which contains the points).

Although the adaptive-voxelization has reduced the number of voxels, the number is still quite large. Thus the almost $o(n^2)$ computing process (Dijkstra Algorithm) is still consuming much time. In order to accelerate the computing process, we propose a novel method. Referring to the main ideas proposed by Barnes and Hut [7] and Gray and More [8], we propose to construct a hierarchical graph instead.

3.1 Construct the Hierarchical Graph

The distance, as a shape feature, is mainly used to describe the attribute of shapes and can reflect the distribution of shape points. As a matter of fact, when the points are close to each other, the distance between them must be very precise; when they are far away, an approximate value can meet the requirement. Thus we propose to construct a hierarchical graph.

We construct several levels of graphs according to the highest resolution level of the shape voxelization results.

The hierarchical graph $G = \{G_0, G_1, \ldots, G_t\} (t \le Res_{max})$. For each graph G_i , $Res_i < (Res_{max} - i)$, Res_i is the maximum resolution revel of voxels which represent

nodes of G_i . We construct the hierarchical graph from G_0 to G_t so that when we construct , we can obtain the correspondence between G_i and G_{i-1} .

The constructing process is shown as follows:

1. Traverse the octree, select all the leaf nodes whose attribute is inside voxel or surface voxel as the graph nodes of G_0 .
2. For each graph G_i , we first traverse all the nodes of G_{i-1} , let r_j be the resolution level of the voxel corresponding to the node j in G_{i-1} .
3. If $r_j < (Res_{max} - i)$, the node j is selected to construct the graph G_i ; else we trace back to the parent of the voxel, then judge the resolution again.
4. Then record the mapping relationship between the nodes in G_i and G_{i-1} .

After finishing all this process, we construct the hierarchical graph successfully.

3.2 Distance Rectification

When the graph becomes coarser, which indicates the maximum resolution level becomes smaller, the error of the inner-distance that we compute will increase. Thus we need to do some rectification so as to get accurate results.

When we analyze the voxelization results, we will find that there are several inside voxels whose resolution levels are lower than the maximum resolution of the shape, a part of the shortest path of a lot of point pairs; and we define them as local centre voxels, as shown in Fig 2.

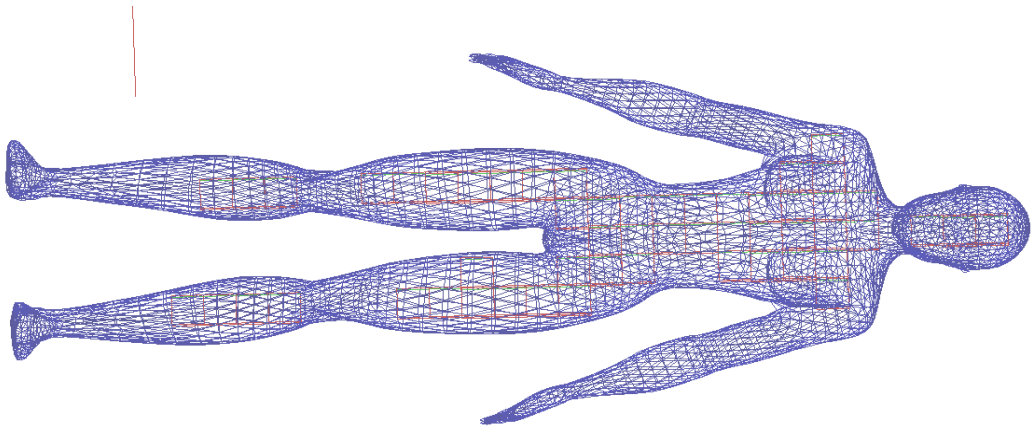

Figure 2 : Some big local center voxels are shown in red cubes in the figure. They are just a part of the local center voxels rather than all the local center voxels.

As the local center voxels are not in a fixed resolution but located at different part of the shapes, we can uniformly select N voxels according to the distribution as the rectifying center V_N , which is much smaller than the number of the voxels. Then we extract the shortest path from voxels in V_N by using the graph G_0 , and record the distance from the local centre voxels to all the surface voxels which contain the sample points; therefore, if we compute the inner-distance between voxels A and B, dis_{AB} , we firstly find the shortest path by using the proper graph, and find local centre voxel C in the path, then replace the distance with the sum of AC and BC, $dis_{AB} = dis_{AC} + dis_{BC}$. In this way, we can ensure the accuracy of the computing results.

3.3 Computing Process

In the hierarchical graph $G(t)$, we extract the shortest path starting from one voxel for t times in each graph and set t thresholds $thres_i(t)$ for each graph. When the distance is bigger

than the $thres_i$, we stop the extracting process, which means that when we use higher level graph whose contains more nodes, we just extract paths containing a little nodes, which will greatly accelerate the almost $o(n^2)$ process of extracting shortest paths. The outline of our algorithm for computing an approximation of inner-distance is given below.

Multi-Level Algorithm for fast computing the inner-distance:

1. Complete the process of adaptive-voxelization.
2. Construct the hierarchical graph $G(t)$.
3. Extract the shortest path and record the distance data from the local center voxels V_N in graph G_0 .
4. For each surface voxel, we extract the shortest path from it in the graph $G(t)$ (proceed the extracting process for each graph G_i), record the path and distance.
5. Proceed the distance rectification for $\{G_0, G_1, \ldots, G_t\}$. When the path includes the local center voxels, we rectify the distance through one of them.
6. For each point pair, we search the distance data from G_0 to G_t , once we find valid data, stop searching and record it as the inner-distance of the point pair.

3.4 Average Inner-Distance

The inner-distance data can reflect the attribute of point pairs. If we want to get the attribute of every point in 3D shapes, we should calculate the average inner-distance [3]. In this case, we don't extract key points; thus every surface voxel equals to a key point. Moreover, in our adaptive-voxelization method, the surface voxels are of the same size. Thus we can simply the calculating equation, which is shown as follows:

$$u_i = \sum_{1}^{N} dis_{ij} \tag{3.1}$$

In this equation, dis_{ij} represents the inner-distance data and N is the number of the surface voxels. To make it stable, we normalize the average inner-distance:

$$un_i = (u_{max} - u_i)/(u_{max} - u_{min}) \tag{3.2}$$

We give some average inner-distance instance in Fig. 3, in which, the color: red, yellow, green to blue represents the data from minimum to maximum.

4. Experiment and Results

In this section, in order to show the performance of our novel method, we will give some results of former methods and ours.

We demonstrate our experiment on several artificial and scanned models, and each model is originally represented in triangle formats. Table 1 gives the time in seconds for the multi-level method in reference to the paper and the method as proposed by Chen and Yan [3], where T1 is the computational time of our method, T2 is the computational time of theirs, n is the number of the points in the models, m is the number of the voxels of the models in volumetric forms, k is the maximum resolution level of the models and A.E. is for average error.

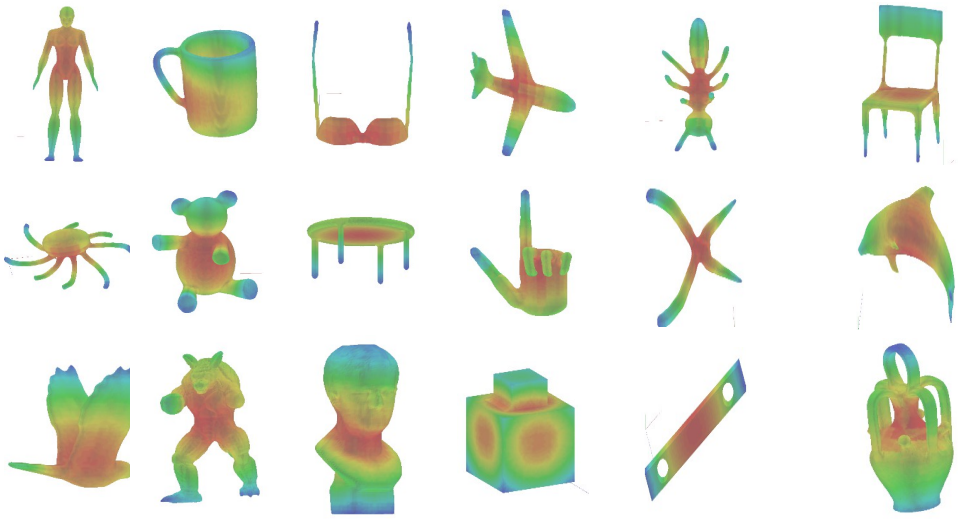

Figure 3 : Some Average Inner-distance Instances

Then we test the accuracy of our method. In the experiment, we compare our results with those generated by the method of Chen and Yan [3] and the one using the fixed resolution graph. We will list the maximum distance error and average distance error (as the method using the fixed resolution graph costs too much time, we only compare some sample point pairs). Considering when the points are nearby, we should get the accurate value; and when the points are far away, we only need the approximate value. We add a weight w when computing the average error according to the distance of point pairs. Experiment results are shown in Table 2. Max error 1 and average error 1 are the results in case of comparison with the method of Chen and Yan [3], and average error 2 are the results in case of comparison with method by using fixed resolution graph.

Model	T1(s)	T2(s)	A.E.(%)	n	m	k
Human	17.654	69.45	3.66	4706	15333	7
Cups	34.077	326.76	4.17	15198	23705	6
Airplanes	2.27	20.21	4.68	6448	4171	6
Ants	5.325	47.27	5.14	8388	6427	6
Chairs	9.54	80.19	4.35	14372	8321	6
Birds	7.899	69.08	6.6	3478	10250	7
Bears	16.429	117.04	5.38	10752	9846	6

Table 1: Inner-distance Computational Time of Different Methods

According to the data in Table 2, the average error is almost 5% and the max error is 20% (the point pairs are far away); when point pairs are nearby, the error is lower than 5%. Thus the results obtained by our method are precise enough to describe the shape feature. In respect of the data in Table 1, the run time of our method is much less than theirs.

Model	Max Error 1	Average Error 1	Average Error 2
Human	21.5	3.66	4.35
Spectacles	18.4	4.25	4.78
Airplanes	23.39	4.68	5.02
Hands	19.45	4.57	5.14
Pliers	24.67	6.41	6.61
Fishes	22.92	5.81	6.02
Birds	16.95	6.6	6.87

Table 2: Inner-distance Computational Accuracy Comparison

5. Conclusion

In this paper, we propose a novel multi-level method of extracting all-pairs 3D inner-distances of various shape models. The experiment results show that, when compared with the former methods, our method can accelerate the computing process greatly. When the data scale grows larger, our method will show better performance in terms of the computing time.

References

[1] H. Ling and D. Jacobs. *Shape Classification Using the Inner-Distance* [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 29(2): 286-299(2007).

[2] Y.-S. Liu, K. Ramani and M. Liu. *Computing the Inner Distances of Volumetric Models for Articulated Shape Description with a Visibility Graph* [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(12): 2538-44(2011).

[3] R. Q. Chen and J. Q Yan. *3D Shape Inner-Distance Computing for Shape Matching Based on Adaptive Volume Representation* [C]. 2014 International Academic Conference on the Information Science and Communication Engineering, 121-126(2014).

[4] X. J. Wu, W. J. Liu and T. R. Wang. *Octree Structure Based Voxelization of Polygonal Meshes* [J]. Journal of Engineering Graphics, 26(4):1-7(2005)(In Chinese) .

[5] B. Mu, J. Deng and M. Pan. *Fast Large Scale Voxelization Using Projection Volume and Octree* [J]. Geography and Geo-Information Science, 26(4):27-31(2010)(In Chinese)

[6] B. Adams and P. Dutre. *Interactive Boolean Operations on Surfel-Bounded Solids*[J]. ACM Transactions on Graphics, 22(3): 651-656(2003).

[7] J. Barnes and P. Hut. *A hierarchical O(NlogN) Force-Calculation Algorithm* [J]. Nature, 324(6096):446-9(1986).

[8] A.G. Gray and A.W. More. *'No-body' Problems in Statistical Learning* [C]. 14th Annual Neural Information Processing Systems Conference, 13:521-527(2001).