Union of Spheres (UoS) Model for Volumetric Data

Vishwa Ranjan and Alain Fournier
Department of Computer Science, University of British Columbia

Introduction

A *stable* representation of an object means that the representation is unique, is independent of the sampling geometry, resolution, noise, and other small distortions in the data, and is instead linked to the shape of the object. Stable representations help characterize shapes for comparison or recognition; skeletal (or medial axis) and volume primitive models have been popular in vision for the same reason. Piecewise polyhedral representations, e.g., tetrahedra, and voxel representations, e.g., octrees, generally tend to be unstable.

We propose a representation for 3D objects based on the set union of overlapping sphere primitives. This union of spheres (UoS) model has some attractive properties for computer graphics, computational vision, and scientific visualization [1, 2]. This communication briefly explains how we can obtain a stable UoS model for objects from *volumetric* data, i.e., data in which we can test if a given point is inside the defined object or outside. This constraint makes the problem of object reconstruction more manageable (Figure 1). In the absence of such inside-outside information, reconstructing an object is a tough problem [3]. For our method, the input is a discrete set of points on the boundary of an object defined in 3D discrete volumetric data such as a CT scan or MRI. The boundary points may be obtained by choosing a threshold (or isovalue) in the data or by other edge-detection methods.

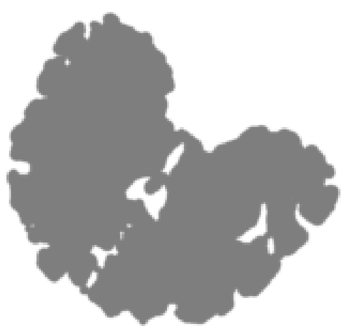

Figure 1: Brain from one slice of MRI head data represented as a union of circles.

The Algorithm

Our method has two steps [4]. In the first step, we generate a UoS representation which is truthful to the input volumetric data in the following sense: (a) all inside voxels are included in at least one sphere, (b) no outside voxel is within any sphere, (c) all boundary points are on the surface of at least one sphere, (d) no boundary point is inside any sphere. In the second step, we simplify this representation to make it stable and to reduce the number of primitives. This allows faster display and facilitates comparison of objects by reducing them to nearly the same number of primitives.

Step I: The dense UoS representation is obtained using the Delaunay tetrahedralization (DT) as follows:

1. Calculate the DT of the boundary points.
2. Compute the circumscribing sphere to each tetrahedron.
3. Use the empty sphere property (ESP) of the DT to verify from the original data which spheres are inside the defined object. This makes use of the fact that any circumscribing sphere contains either only outside voxels or only inside voxels. This is true because if any sphere contains both inside and outside voxels, it will contain a boundary point as well; and this is forbidden by the ESP of the DT.

Spheres generated in this fashion satisfy all the truthfulness criteria previously mentioned. The properties of the sphere representation are closely linked to the properties of the underlying tetrahedra representation. Whereas the method is easy to implement, the worst case complexity of the algorithm is $O(n^2)$ in both time and number of primitives, where n is the number of boundary points. However, the highest number of primitives observed in our experiments so far has been about $4n$.

Step II: We simplify the dense sphere representation as generated above by clustering similar or close-by spheres and eliminating redundant or non-significant spheres. Many point clustering algorithms could be used. We defined a new clustering method which uses the fact that the underlying primitives are spheres and not points, and gives a better handle on the error introduced during simplification. It is based on the concept of *sphericity* which is a measure of how well a group of spheres can be represented by a single sphere. Sphericity can be defined as the ratio of the volume of union of spheres in the group to the volume of the smallest bounding sphere. For computational convenience, we define sphericity as the ratio of the radius of the largest sphere in the group to the radius of the smallest enclosing sphere. In either case, sphericity is a number between 0 (very non-spherical) and 1 (perfectly spherical).

The simplification algorithm processes the spheres from the largest to the smallest. In every iteration, using the largest remaining sphere, the algorithm calculates a new sphere such that the sphericity $\geq \sigma$ (a user defined threshold, generally close to 1), and it encloses a maximum number of remaining spheres. All the spheres within the newly created sphere are then deleted and the iterations continue until all the spheres in the original representation have been taken care of. The time complexity of the algorithm is $O(n^2)$, n being the number of spheres in the representation to be simplified.

We have used the algorithm to simplify a variety of objects. It generally yields a stable representation and can be applied with different values of sphericity threshold σ. In the extreme cases, when $\sigma = 0$, we get a single sphere as the output, and when $\sigma = 1$, we get the original representation itself as the output. For any intermediate value of σ, we get some intermediate number of spheres in the output. Since the algorithm ensures that all the generated spheres have a sphericity $\geq \sigma$, it puts a bound on the simplification error. However, like most clustering algorithms, it does not guarantee preservation of topology.

Application to Object Registration

We used the stability property of the UoS representation for the problem of object registration: Given two representations, determine if they represent the same object, and if they do, find the transformation from one to the other. To register objects, we first match a sufficient number of spheres from the UoS representation of one object to the other, and then from the matches find the most likely transformation using the method of least squares. The matches between the spheres of two representations A and B are obtained by formulating the problem as a minimum weight bipartite graph matching (also known as "assignment") problem. First, we assign distances from every sphere $a \in A$ to every sphere $b \in B$ according to a pre-defined metric (which is a function of the size of the sphere, and its location with respect to the coordinate frame determined from principal moments). Next, we do a minimum weight matching on the resulting bipartite graph. This problem has been widely studied and can be solved exactly in $O(n^3)$ time, where n is the number of nodes (in our case, the number of spheres in the simplified representation) in the bipartite graph. The results on the experimental data look promising. To determine matches in a bipartite graph with about 400 nodes in total, an IRIS Crimson takes about 1.5 seconds.

Implementation

The described algorithms have been implemented (for 2D and 3D) in the C language and compiled for SGI and IBM RS/6000 machines. Some of the results are shown in images accessible through the URL address `<http://www.cs.ubc.ca/spider/ranjan/>`.

Conclusion

Union of spheres (UoS) representation has many desirable properties for graphics, vision, and visualization. In this communication, we described how we can generate and simplify a UoS representation for volumetric data. The derived representation can be used to characterize and compare shapes; we applied it to the problem of object registration and the results are encouraging.

Acknowledgements

We gratefully acknowledge the help of the Geometry Center for the triangulation code, Bernd Gaertner for the code for finding the smallest disk enclosing a set of points, and Robert Kennedy for the code for solving the assignment problem.

References

[1] F. Aurenhammer. "Power diagrams: properties, algorithms and applications", *SIAM Journal of Computing*, Vol. 16, 1987, pp. 78–96.

[2] H. Edelsbrunner. "The union of balls and its dual shape", *Proceedings of the 9th Annual Symposium on Computational Geometry*, 1993, pp. 218–229.

[3] H. Edelsbrunner and E. P.Mücke. "Three-dimensional alpha shapes", *ACM Transactions on Graphics*, Vol. 13, No. 1, January 1994, pp. 43–72.

[4] V. Ranjan and A. Fournier. "Volume models for volumetric data", *IEEE Computer*, Vol. 27, No. 7, July 1994, pp. 28–36.

[5] E. Welzl. "Smallest enclosing disks, balls and ellipsoids", Tech. Report No. B 91-09, Fachbereich Mathematik, Freie Universität Berlin, Berlin, Germany, 1991.