SIGRAD 2015
L. Kjelldahl and C. Peters (Editors)

Exact Bounding Spheres by Iterative Octant Scan

Thomas Larsson

School of Innovation, Design and Engineering
Mälardalen University, Sweden

Abstract

We propose an exact minimum bounding sphere algorithm for large point sets in low dimensions. It aims to reduce the number of required passes by retrieving a well-balanced set of outliers in each linear search through the input. The behaviour of the algorithm is mainly studied in the important three-dimensional case. The experimental evidence indicates that the convergence rate is superior compared to previous exact methods, which effectively results in up to three times as fast execution times. Furthermore, the run times are not far behind simple 2-pass constant approximation heuristics.

Categories and Subject Descriptors (according to ACM CCS): F.2.2 [Analysis of algorithms and problem complexity]: Nonnumerical Algorithms and Problems—Geometrical problems and computations; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems

1. Introduction

The bounding sphere appears to be a ubiquitous tool for creating coarse conservative representations of more detailed and complicated geometric objects, such as polygon meshes and point clouds. The spheres are often arranged in a hierarchical data structure, the sphere tree, that provides a powerful representation of a complex object at various levels-of-detail [BO04]. Numerous applications in optimization, geometry, computer graphics, simulation, and games bear witness of the popularity of the bounding sphere. Not surprisingly, the optimal bounding sphere is known by many names, including minimum enclosing ball (MEB), minmax location, 1-center, and minimum covering sphere. To find it, we need to determine the point that minimizes the maximum distance to the input elements [EH72].

The advantages of using the sphere as a container stem from its simple shape, which gives a low storage cost and makes geometrical operations uncomplicated and fast. On the other hand, the main criticism concerns the quite poor fit it provides of the enclosed objects, which under certain difficult conditions makes the sphere inappropriate as a bounding volume [GLM96]. Even so, by combining the sphere with some other shape to provide a more flexible and finer approximate representation, much of the beneficial features of the sphere may still be exploited also in difficult scenarios. Such combinations have been utilized successfully to speed up neighbour queries [KS97], ray tracing [WHG84], and collision detection [LAM09, CWK10].

Of course, to support bounding spheres, an appropriate construction algorithm is needed. It is straightforward to use a simple constant approximation method, which given n points in dimension d scans the input one or two times [ZZC06, Rit90]. However, since the main deficiency of the bounding sphere has to do with its size, any excessive space may be troublesome. The goal of this paper is to propose an algorithm that allows minimum bounding spheres to be computed rapidly enough for real-time computer graphics and simulation applications.

It is well-known that the MEB of a point set $P = \{p_0, p_1, \ldots, p_{n-1}\} \subset \mathbb{R}^d$ is unique and defined by a support set of up to $d+1$ points on its surface. Most previous exact methods target low-dimensional instances of the problem. A notable exception is the procedure given by Fischer et al. [FGK03]. Otherwise, computing the exact minimum sphere is doable in expected linear time in fixed dimension using Welzl's randomized algorithm [Wel91]. Although it has been demonstrated that this approach can be accelerated in practice by move-to-front and pivoting heuristics [Gär99], it may still require quite many passes over the input, and the computation of intermediate solutions may also involve solving numerous primitive cases to find valid support sets.

In the important three-dimensional case, we can use a more balanced way of retrieving *outliers*, i.e., points located outside the intermediate solutions, which is based on subdividing the space into octants during the linear scans of the input. In this way, the number of required passes can be reduced considerably, which in turns leads to fewer computations of intermediate solutions. Note that the same principle can be utilized more generally in dimensions beyond the third by subdividing the linear searches into hyper-octants, also known as orthants. However, their number grows exponentially with the dimension.

Based on these observations, we propose an exact MEB algorithm with an improved convergence rate and faster run time. Experimental validation confirms its efficiency and robustness in the three-dimensional case.

2. Algorithm

Conceptually, the algorithm is very simple. An initial minimum sphere is defined around a constant number of input points. The input is then scanned to find the farthest outlier in each octant using the current center as the origin for the search, which gives up to eight outliers in each pass. The tentative sphere is then updated to the next intermediate solution by taking these outliers in consideration. This procedure is repeated iteratively until no outliers can be found, meaning that the smallest possible sphere with a valid support set has been found. We refer to this approach as the *Iterative Octant Scan* algorithm.

This technique of incrementally enlarging the sphere and maintaining a valid support set follows Gärtner's approach [Gär99]. Since P is a finite set and the radius is monotonically increasing without ever exceeding the wanted minimum radius, the iterations can be repeated until the exact minimum sphere is obtained. In his approach, however, only "the most promising" point, i.e., the point with largest distance from the current center, is added in each iteration. By collecting a more complete view of the remaining outliers in each scan, the convergence rate can be improved. The following pseudocode describes the algorithm at a high level:

```
ITERATIVEOCTANTSCANMEB($P$)
1.  $S, c, r \leftarrow$ SOLVESAMPLEDMEB($P$, $m$)
2.  **repeat**
3.      $Q, h \leftarrow$ FARTHESTOCTANTPNTS($c, r, P$)
4.      **if** $Q = \emptyset$ **then exit loop**
5.      $S \leftarrow S \cup Q$
6.      $S, c, r \leftarrow$ SOLVEMEB($S$)
7.  **return** $c, r$
```

The algorithm starts off by defining a MEB with center c and radius r around a constant number of points drawn randomly from the input set $P = \{\, p_0, p_1, \ldots, p_{n-1} \,\}$. When n is large, we use the constant $m = 20$. The candidate support set S is also initialized with the supporting points of this initial MEB. Although it would be enough to base the initialization on a single input point, say by $c \leftarrow p_0$, $r \leftarrow 0$, and $S \leftarrow p_0$, the sampled MEB approach generally helps in reducing the number passes, without introducing much overhead.

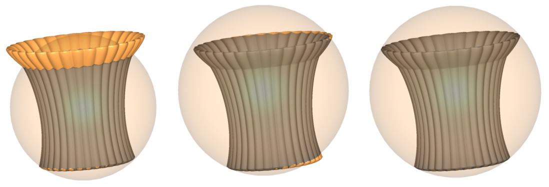

Figure 1: *Illustration of the rapid progress of the iterative octant scan approach when applied on the Flowerpot mesh. The left image shows the initial sampled ball. Then the middle and right images show the intermediate solutions found in iteration 1 and 2, respectively. In iteration 3, a slightly larger ball is found (which looks just like the right image). Finally, iteration 4 simply detects that this is the final MEB.*

Then the main loop is executed (Lines 2–6). On Line 3, a simple linear scan of the input is performed. The subroutine locates a set Q with up to eight extremal outliers, one from each octant, using c as the origin. Here extremal outlier refers to the input point that maximizes the Euclidean distance $\|p_i - c\| = \sqrt{(p_i - c) \cdot (p_i - c)}$ per octant. However, only points p_i satisfying $\|p_i - c\| > r$ are possible candidates for being extremal in any one of the octants. This condition is used to effectively filter the points, and only actual outliers are examined in more detail with respect to the octants they belong to. In fact, this linear scan is highly efficient with an almost negligible overhead compared to the standard approach of only finding the actual farthest outlier, which of course is always included among the octant points selected here.

Given that no outlier is found, the sought MEB has been found. This condition for termination is tested on Line 4. Otherwise, the set of outliers Q is added to the candidate support set S (Line 5), and the next intermediate solution is found by calling a subsidiary MEB solver that is able to handle the "low-level" primitive operations efficiently (Line 6). For more details, see Gärtner's treatment of how to robustly solve the primitive operations under floating point arithmetic [Gär99]. After this the main loop repeats until the exact solution is obtained. Figure 1 shows rendered images that illustrate the fast convergence of the algorithm in a specific case.

Clearly, S is a superset of the actual support set S' that defines the intermediate solutions. It is possible to reduce the size of S by assigning $S \leftarrow S'$ in the solver, which ensures $|S| \leq d+1$ at the beginning of the next iteration. However, we have chosen to keep them around, since this can speed up the convergence for certain inputs, and the overhead of this appears to be insignificant for large n in 3D.

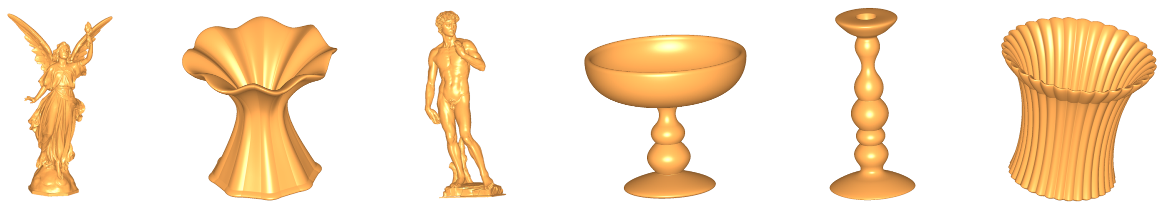

Figure 2: *Some of the triangle meshes used to benchmark the algorithms.*

		Octant Scan MEB			Farthest Point MEB			Ritter's method		
Mesh	n	k	r	t (ms)	k	r	t (ms)	k	r	t (ms)
Lucy	14,027,872	3	872.73962	80	6	872.73962	150	2	966.03467	60
Vase	4,609,442	4	13.769305	43	9	13.769305	78	2	16.842354	19
David	4,129,614	4	2713.4316	34	6	2713.4316	47	2	2842.9419	18
Goblet	1,008,772	4	15.644787	9.0	13	15.644787	24	2	18.267401	4.1
Blade	882,954	2	334.75421	3.8	3	334.75421	5.9	2	345.73364	3.8
Candlestick	456,482	3	37.304390	2.7	6	37.304390	5.0	2	39.363644	1.9
Flowerpot	410,482	4	26.173567	3.2	7	26.173563	5.0	2	32.095146	1.7
Hand	327,323	2	3.508905	1.2	4	3.508905	2.2	2	3.508966	1.4

Table 1: *Minimum bounding sphere computation for n points in 3D. For each tested algorithm, the number of passes k, the resulting radius r, and the sequential run time t in milliseconds (ms) are listed.*

Finally, we note that it is trivial to modify the termination criterion on Line 4 of the exact algorithm to create an alternative algorithm that reports $(1+\varepsilon)$-approximate solutions for any given $\varepsilon > 0$. The current farthest point distance h from c is already known by Line 3. Given that $h \leq (1+\varepsilon)r$, the requested approximation is fulfilled and we can simply return the center c and radius h. This alternative may be used to speed up the computations, depending on the context.

3. Experiments

The presented algorithm was implemented in C++ and compiled in Visual Studio 2013. All tests were performed on a PC with a 2.80 GHz Intel Core i7-4810MQ CPU and 32 GB RAM under Windows 7. The input set was stored in a flat array of 3D points using single precision (32 bits) floating point numbers to represent the coordinates. The auxiliary MEB solver, however, used double precision to produce the intermediate solutions. All runs were executed single-threaded without vectorization. Note, however, that the dominating operation, the retrieval of the octant points with the largest distance from the current center, is susceptible of parallelization, an opportunity which we leave for future exploration (see some further comments about this in Section 4).

To challenge the proposed Octant Scan MEB method, we benchmarked it against two other well-known reference algorithms. The first competing strategy, called Farthest Point MEB, was basically Gärtner's algorithm [Gär99]. To make a fair comparison, however, we used our own adaptation, which shared as much source code as possible with the proposed algorithm. In this way, it also benefited from the same effective initialization of the first intermediate solution based on sampling. In fact, the only difference is that it calls a standard farthest point routine instead of searching for the farthest octant points. The second competing strategy was Ritter's algorithm, which is extremely fast since it only uses two simple passes [Rit90]. The correctness is ensured by overestimating the radius using a greedy update strategy in the final pass (which is also used in [ZZC06]).

A set of complex triangle meshes were chosen as benchmark problems. Several of the used meshes are well-known models from the Stanford 3D Scanning Repository and the Large Geometric Models Archive at Georgia Tech. The other used data sets will also be made publicly available. Rendered images of some of the included meshes are shown in a Figure 2. We measured the number of passes k counted as the number of linear scans through the input points performed by the algorithm as well as the execution time t in milliseconds. The results are given in Table 1.

The experimental results indicate that the iterative octant scan method finishes in fewer passes than the competing exact method in each case. As shown by the Goblet model, the difference can be more than a factor of 3 (which led to a speedup of 2.7). The reported minimum radii were consistent between the two methods. Only very minor differences due to floating point rounding errors have been observed (as in the case of Flowerpot).

As expected, Ritter's approach, with its fixed two passes, gave a more predictable performance. In most cases, it had the fastest execution, but for the Hand model our exact method was actually faster. Overall, the speedup for Ritter versus the exact octant scan method was in the range of 0.9–2.3. However, Ritter's method overestimated the radius by a factor of 1.0–1.23 for the included test cases. This may be reasonable in certain applications, where exact solutions are not mandatory, but note that a radius increased by a factor of $\sqrt[3]{2} \approx 1.26$ doubles the volume in 3D.

To examine the correctness and robustness of our implementation, a brute force testing procedure was followed. Millions of random inputs with $n \in [10, 100]$ points in a cube were tested. In each case, it was confirmed that all points reside in the returned sphere (we used a small tolerance to account for minor rounding errors). The computed ball was also compared to the one given by Gärtner's publicly available miniball software. This testing procedure provoked no errors apart from very small differences in the returned radii apparently resulting from minor floating point rounding errors in the exact solver.

4. Conclusions

The proposed algorithm is able to find optimal bounding spheres of large point sets in just a few passes in the three-dimensional case (2–4 simple scans are often sufficient). Its usage of only k calls to a MEB solver on small subproblems to get the intermediate solutions further contributes to its high performance. To the best of our knowledge, this is the first exact algorithm with an execution speed that is almost comparable to that of naive constant approximation heuristics. In fact, when we ran the $(1+\varepsilon)$-approximate version of our algorithm with $\varepsilon = 10^{-2}$, it was faster than Ritter's method in five out of the eight test cases, while it still produced smaller spheres in all eight test cases.

In the future, we would like to extend this paper by including more variants of the algorithm and by considering the behaviour more generally in low dimensions, say when $d \leq 10$. The same basic procedure can be applied, but in dimension d, there are 2^d hyper-octants to consider. Of course, this exponential growth seems problematic, but for large data sets, the approach may still be useful beyond the three-dimensional case. Additional filtering heuristics can also be designed to improve the performance, for instance by exploiting inherent spatial coherence in the data sets.

Furthermore, the possibilities for parallelizing the computations look compelling (cf. [KWL10, KL14]). The iterative octant scan method seems amenable for modern architectures supporting hierarchical levels of parallelism. By using a divide and conquer strategy, vectorized local scans can be performed on subsets of the input, and the results merged using a suitable reduction technique. In the end, this would hopefully lead to a fast and scalable data-parallel solution. This is definitely needed for massive data sets, such as scanned point clouds, which may contain billions of points.

Acknowledgements

The author is supported by the Swedish Foundation for Strategic Research through grant no. IIS11-0060. The models Lucy and David were kindly made available by the Stanford Computer Graphics Laboratory, the latter through the Stanford Digital Michelangelo Project Archive. The Blade and Hand models were provided by the Large Geometric Models Archive at Georgia Tech.

References

[BO04] Bradshaw G., O'Sullivan C.: Adaptive medial-axis approximation for sphere-tree construction. *ACM Transactions on Graphics 23*, 1 (Jan. 2004), 1–26. 1

[CWK10] Chang J.-W., Wang W., Kim M.-S.: Efficient collision detection using a dual OBB-sphere bounding volume hierarchy. *Computer Aided Design 42*, 1 (2010), 50–57. 1

[EH72] Elzinga J., Hearn D.: The minimum covering sphere problem. *Management Science 19*, 1 (1972), 96–104. 1

[FGK03] Fischer K., Gärtner B., Kutz M.: Fast smallest-enclosing-ball computation in high dimensions. In *In Proceedings of the 11th Annual European Symposium on Algorithms (ESA)* (2003), Springer-Verlag, pp. 630–641. 1

[Gär99] Gärtner B.: Fast and robust smallest enclosing balls. In *Proceedings of the 7th Annual European Symposium on Algorithms* (1999), Springer-Verlag, pp. 325–338. 1, 2, 3

[GLM96] Gottschalk S., Lin M. C., Manocha D.: OBB-Tree: a hierarchical structure for rapid interference detection. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), pp. 171–180. 1

[KL14] Källberg L., Larsson T.: Accelerated computation of minimum enclosing balls by GPU parallelization and distance filtering. In *Proceedings of SIGRAD 2014* (June 2014), pp. 57–65. 4

[KS97] Katayama N., Satoh S.: The SR-tree: an index structure for high-dimensional nearest neighbor queries. In *Proceedings of the 1997 ACM SIGMOD international conference on Management of data* (1997), ACM, pp. 369–380. 1

[KWL10] Karlsson M., Winberg O., Larsson T.: Parallel construction of bounding volumes. In *Proceedings of SIGRAD 2010* (November 2010), pp. 65–69. 4

[LAM09] Larsson T., Akenine-Möller T.: Bounding volume hierarchies of slab cut balls. *Computer Graphics Forum 28*, 8 (2009), 2379–2395. 1

[Rit90] Ritter J.: An efficient bounding sphere. In *Graphics Gems*, Glassner A., (Ed.). Academic Press, 1990, pp. 301–303. 1, 3

[Wel91] Welzl E.: Smallest enclosing disks (balls and ellipsoids). In *New Results and New Trends in Computer Science*, Maurer H., (Ed.), vol. 555 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 1991, pp. 359–370. 1

[WHG84] Weghorst H., Hooper G., Greenberg D. P.: Improved computational methods for ray tracing. *ACM Transactions on Graphics 3*, 1 (1984), 52–69. 1

[ZZC06] Zarrabi-Zadeh H., Chan T. M.: A simple streaming algorithm for minimum enclosing balls. In *Proceedings of the 18th Canadian Conference on Computational Geometry* (2006), pp. 139–142. 1, 3