The 5th Annual IEEE International Conference on
Cyber Technology in Automation, Control and Intelligent Systems
June 8-12, 2015, Shenyang, China

Bounding Sphere CAD Model Simplification for Efficient Collision Detection in Offline Programming

Alex Visser, Zengxi Pan, Stephen van Duin

***Abstract*—Collision detection performance is one of the major bottlenecks of Automated Offline Programming (AOLP). Simplifying CAD models allows collision detection algorithms to run more quickly, as a result more complex procedures are able to be calculated in a reasonable amount of time using AOLP. This paper presents a randomised bounding volume method for calculating a sphere based representation of a CAD model while conservatively removing small features. The output has fewer spheres when compared with other sphere packing techniques due to the first stage of the algorithm which removes small features from the model. Using the representation generated by the bounding sphere simplification algorithm for collision detection can dramatically improve performance in simulations and motion planning processes commonly used in AOLP.**

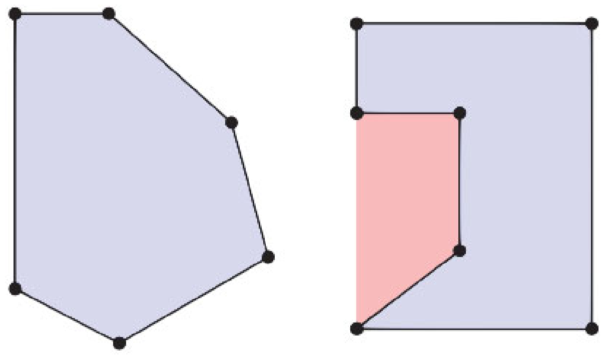
Figure 1: Convex (left) and concave (right) shapes.

I. Background

Triangle meshes are widely used to represent objects in computer programs. Triangles are easily drawn by computers and a mesh can represent any arbitrary shape provided enough triangles are used.

Triangle meshes are also used in computer graphics for collision detection. Triangle meshes do not provide an ideal representation of models to be used in collision detection because there is generally no simple mathematical formula that describes the shape of the object such as is the case with spheres, cubes and other simple shapes. This means that triangles often must be individually checked to determine if there is an intersection with other triangles which takes a significant amount of time as the complexity of a model increases.

Mesh representations of objects that are used for visualisations are generally simplified for the purpose of being used in collision detection when they are used in real world applications. Model simplification is very important in applications such as computer games where performance is critical.

Simplifying a CAD model by definition means that the geometry of the output will be less accurate than that of the input. This loss of accuracy is not critical for many applications such as computer games where collisions are only required to provide enough realism. Applications such as AOLP however, require a high level of accuracy as robots, specifically their end effectors often need to be within millimetres of other objects. In this case the geometry needs to be modelled very accurately around the sections that are required to interact closely with other objects.

Triangle meshes can be categorised into two general types for the purpose of collision detection. Convex meshes and concave meshes. Convex meshes are a representation of a completely convex shape. Collision detection algorithms can take advantage of convex shapes because moving along a convex mesh in one direction means the vertices will either move closer or further away from any arbitrary point until a global maximum or minimum distance is reached. Concave meshes contain at least one concave section and when moving along a concave mesh from vertex to vertex there are local maximums and minimums to any arbitrary point. As a result, convex meshes performs much better than concave meshes in collision detection and it is desirable to create convex shapes to use in collision detection.

Primitive objects that include spheres, rectangular prisms and cylinders have the best performance in collision detection algorithms because they are convex shapes that can be described mathematically. The issue with using primitives in collision detection is that they do not accurately represent many real objects and representing most objects with a single primitive shape will yield poor collision detection results. One solution is to use a collection of primitive objects to represent a more complex object. This can either be done manually, as is often the case in computer graphics or automatically as shown in literature [1], [2].

The best performing collision objects that still retain a high level of accuracy are manually created because a human can balance performance with expected accuracy as they simplify a model. Unlike humans, simplification algorithms do not contain information on the required accuracy of different parts of a model. Processing CAD models is costly because high quality simplifications must be manually created. The benefit is very fast and accurate performance. Unless the cost of manually simplifying a CAD model is able to be recovered in the form of time savings, achieved by using the simplified model, it is not economically feasible to manually simplify a CAD model.

Several algorithms have been presented in literature that automate the simplification of triangle meshes; these include decimation based methods [3], [4], envelope methods [5] and decomposition methods [6], [7]. These algorithms are all based on different objectives for the simplified shape. Because there are different objectives for each of these algorithms a suitable algorithm must be picked based on the input model and the objective of the simplification otherwise the desired simplification may not be achieved. Often simplification algorithms are developed for visual simplification of CAD and are not suited for use in collision detection as is the case in [3] and [8].

AOLP is the process of generating robot programs automatically from a model of a workspace. The general procedure involves two processes that perform a significant amount of collision checks. The first is planning process points for a particular operation such as painting or welding, this requires multiple tool angles and robot configurations to be examined to determine a continuous motion that allows the robot to complete the operation successfully. The second is planning a collision free path between the start and end of the process to the home position of the robot.

AOLP requires that collision checking is highly accurate and very quick. Most simplification algorithms are a trade off between accuracy and performance. Usually AOLP deals with mechanical CAD models which are very highly detailed. The high level of detail often present in CAD models is not required for use in motion planning for industrial robots. Often the interior parts of a model, particularly a mechanical CAD model, are not reachable by any device, as a result it does not make sense to keep these additional details in a model that will be used for collision detection in motion planning algorithms. In AOLP, a false negative during collision detection has the potential to allow code to be generated that causes the robot to crash into a real world object. Therefore it is important to avoid any false negatives in collision detection, even if that results in false positives being created.

The rest of this paper is organised as follows. Section II presents the bounding sphere simplification algorithm, Section III compares the algorithm with other commonly used mesh simplification algorithms and Section IV covers the performance and accuracy of CAD representations produced by the algorithm.

II. Algorithm

This section presents the bounding sphere simplification algorithm which simplifies CAD models into a collection of spheres. The algorithm first performs a simplification stage where small features are removed conservatively. This simplification is achieved by bounding the original CAD model with spheres of a minimum size on the exterior of the model. Sections of the model that these external spheres are not able to cover will become part of the output CAD. The minimum size of the spheres is ideally based on other geometry that is going to interact with the model. For example, in a robotic welding environment, the external sphere size could be set to the diameter of the smallest object in the simulation likely to be involved in collisions, possibly the gas shield of a weld gun. This would result in the removal of all features which are smaller than the gas shield, such as screw holes.

The second phase of the algorithm performs sphere fitting on the volume bound by the external spheres. In this paper, a generic sphere packing algorithm is used but more advanced algorithms such as Adaptive Medial Axis Approximation (AMAA) [1] could be used and theoretically result in further reductions of the number of spheres created.

Figure 2 shows the advantage of using the bounding sphere simplification algorithm over a sphere packing method to fit spheres when the input model contains concave sections. A similar level of error is achieved around the outer edges using a third of the spheres.

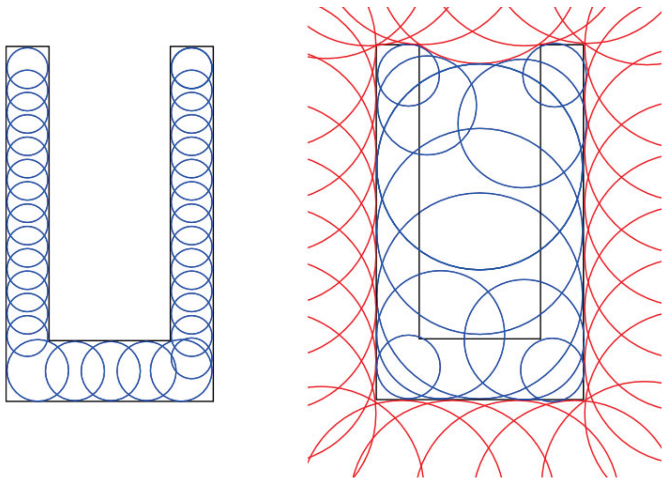
Figure 2: Sphere packing (33 spheres, left) and bounding sphere simplification (11 spheres, right).

Details of the bounding sphere simplification are shown in Algorithm 1. There are two variables that define the performance of the algorithm, the minimum size of the external spheres and the maximum error allowed in the final representation.

A. Minimum External Sphere Size

The minimum size of the external spheres (min_ext_size in Algorithm 1) defines the level of simplification that is applied to the CAD model. This variable should be chosen based on the size of objects that will be colliding with the obstacle. If a size of 100 mm is chosen, concave sections of a mesh smaller than 100 mm will be removed from the model. Figure 3 shows a graph of the amount of space occupied by the external spheres as the minimum external sphere size is increased. The model used in Figure 3 is a hollow pipe with a diameter of 40 mm. When the minimum external sphere size starts to reach 40 mm, the external spheres are not able to fit inside the pipe and the internal section of the pipe will be simplified out of the final representation. For example, when simplifying objects that will be colliding with a mallet, a suitable minimum external sphere size would be equal to the diameter of the handle of the mallet. This would result in retaining the geometry of the concave sections of the objects that the hammer could penetrate. All of the concave sections of the model that are too small for the mallet to penetrate would be removed.

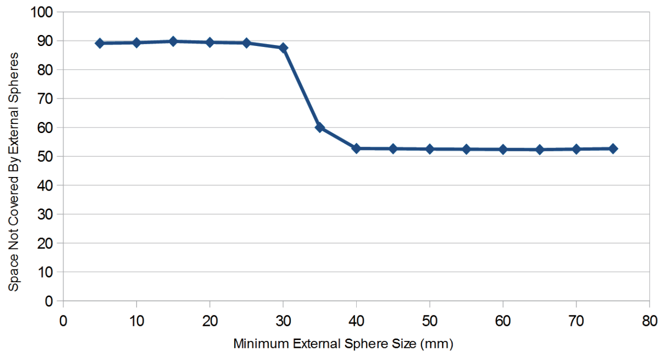

Figure 3: Percentage of space not covered by external spheres as the minimum size of the external spheres is increased.

B. *Maximum Error*

The maximum error that is allowed in the final sphere representation (error_limit in Algorithm 1) has a significant impact on the output of the algorithm. This error is the maximum distance from furthest point bound by the external spheres to any of the internal spheres. As this variable is reduced the accuracy of the representation increases and the amount of spheres required in the representation also increases. To guarantee full coverage of the original CAD model by the sphere representation the radius of every sphere in the representation can be increased by the maximum error.

III. Commonly Used Simplification Methods

Figure 4 shows a comparison of the output of a 4 different simplification algorithms. Each of the algorithms simplify the model differently and have their own ideal use cases.

Decimation : Triangle mesh decimation is a method which reduces the amount of triangles in a mesh while preserving the topology as much as is possible [3]. Decimation of a triangle mesh is one of the most obvious ways to simplify a shape, however unless the input model is over sampled decimation can only provide relatively small performance

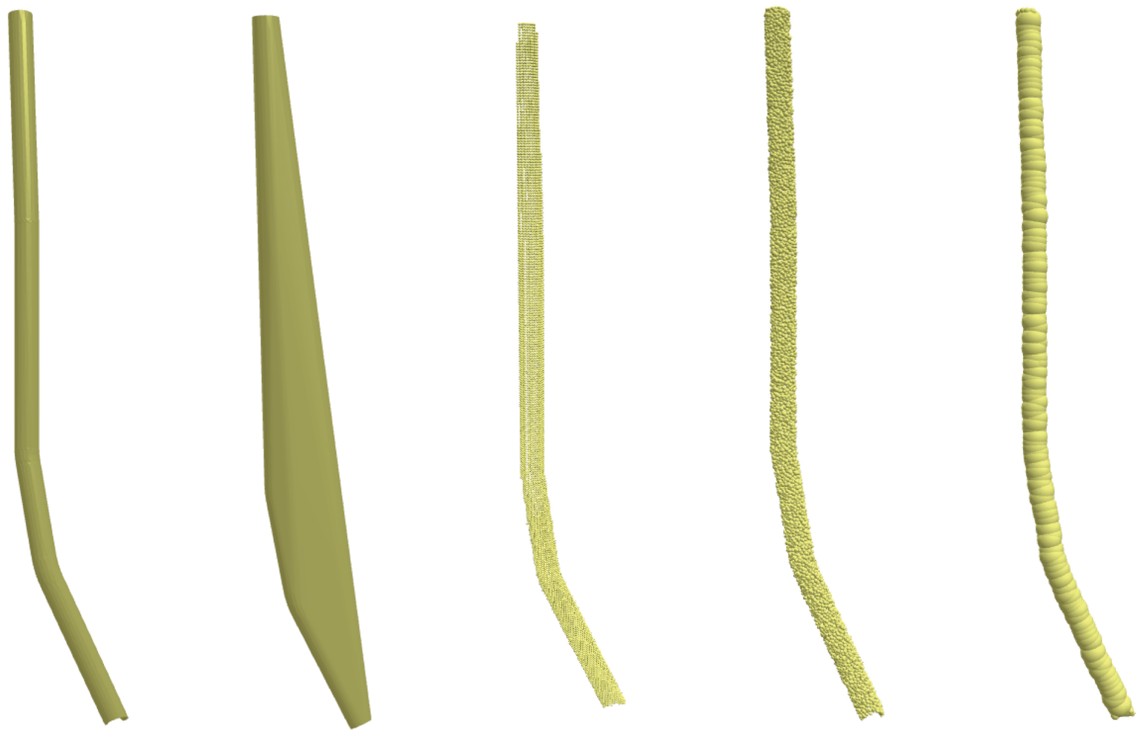

Figure 4: Different representations of a pipe shaped model. (a) original mesh, (b) convex hull, (c) voxelisation, (d) sphere packing, (e) bounding sphere simplification algorithm.

Algorithm 1 Bounding sphere simplification algorithm.

```
while external sphere count < limit do
  add external sphere $S_e$ at random position
  set $S_e$ diameter to min_ext_size
  if $S_e$ is in collision then
    remove $S_e$
  else
    while $S_e$ is not in collision do
      expand $S_e$
    end while
  end if
end while
remove original triangle mesh
max_error = $\infty$
max_error_pt = random point
while max_error < error_limit do
  add internal sphere $S_i$ at max_error_pt
  set $S_i$ diameter to min_internal_size
  if $S_i$ collision then
    remove sphere
  else
    while $S_i$ ! colliding in two or more places do
      if $S_i$ colliding in one place then
        move $S_i$ away from the collision along the normal
      else
        expand $S_i$
      end if
    end while
  end if
  max_error = distance from furthest point bound by the external spheres to any of the internal spheres
  max_error_pt = point where max error occurs
end while
```

increases before the original shape becomes too deformed to provide accurate collision detection results. There are several decimation processes which use different heuristics and simplification techniques found in literature [3], [4].

Convex Hull : A convex hull is the smallest convex shape that includes all of the original CAD model. The Quickhull algorithm [5] is able to efficiently and quickly create a convex hull from a set of vertices of a triangle mesh. A convex hull of a shape is more efficient than the original shape in collision detection if the original shape is concave. This is mainly because convex shapes are able to take advantage of heuristic based collision detection algorithms such as the GJK (Gilbert Johnson Keerthi) algorithm [9]. A convex hull of a concave object can significantly change its geometry and make it unsuitable for collision detection, as is the case in Figure 4b.

Approximate Convex Decomposition : The convex decomposition approach creates a set of convex hulls that exactly match the geometry of an object. Alternatively, approximate convex decomposition [6], [7] creates a set of convex hulls that approximate the original shape closely. The original shape can then be represented more accurately compared with a single convex hull, and the accuracy is increased by allowing more convex shapes to be added to the model. Because the shapes produced are all convex, collision detection procedures can take advantage of the GJK algorithm as is the case with a single convex hull.

Voxelisation : Voxelisation creates a grid of voxels, typically cubes or spheres which represent an object [10], [11]. If part of the original mesh lies within a grid section, a voxel is placed into that grid position otherwise it is left empty. Because voxelisation relies on a fixed size voxel the resulting shape is generally far from optimal for collision detection as it results in either many small voxels that represent the model accurately but perform slowly in collision detection due to the number of voxels, or there are large voxels which do not accurately represent the model.

Sphere Packing : Sphere packing involves placing several spheres inside the bounds of a CAD model. There are several different types of sphere packing algorithms [12], [13], [14] which can be classified as allowing or disallowing overlapping spheres and statically or dynamically creating the representation. The algorithm presented above relies on static sphere packing that allows overlapping. Currently, one of the best methods for sphere packing, specifically for use in collision detection, is AMAA, which dynamically creates a sphere representation allowing overlapping. This allows fewer spheres to represent a model when compared with previous sphere packing algorithms. Sphere packing is not suited to models which are long, wide and narrow such as a thin plate because too many spheres are created which can result in worse collision detection performance than the original CAD.

Sphere Trees: A CAD model can be represented by a tree of spheres, each level of the sphere tree approximates the cad model to a higher level of accuracy. Making the sphere tree hierarchical so that each parent sphere fully encompasses its children allows specialised hierarchical collision detection algorithms to be used [15]. These algorithms recursively calculate collisions on each of the nodes of the tree, reducing the total number of objects checked in a collision detection operation. The finest level of detail of a hierarchical sphere tree is very similar to the spheres obtained using sphere packing. Without using hierarchical collision detection algorithms, the collision detection performance achieved using sphere tree models is similar to those achieved with a sphere packing method. The performance improvements associated with using sphere trees come from the algorithms used to implement the collision detection rather than the simplification of the CAD models. As a result, any simplification method that produces sphere based representations of a model, such as the bounding sphere simplification algorithm, can take advantage of these collision detection algorithms by generating a sphere tree from its output.

None of these methods take into account the objects to be

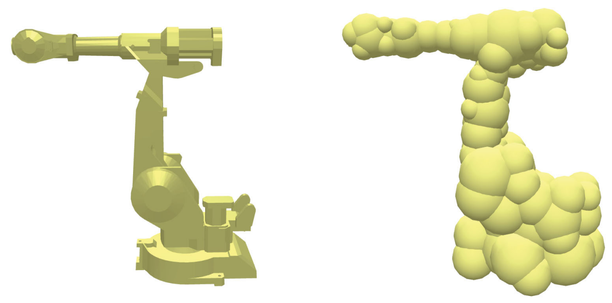

Figure 5: Simplification of an IRB4400 using the bounding sphere simplification algorithm.

used in the collision detection algorithm. Very small features which are commonly found in mechanical CAD drawings, such as mounting holes, affect the simplification that these algorithms perform and cause the output to be more complex than required. A human modeller would remove these small features when creating a collision model as they are unlikely to have a significant effect on the accuracy of collision detection. Some work has gone into algorithms that simplify small features automatically [16] but they have limitations on what features can be removed. The bounding sphere simplification algorithm presented in this paper simplifies small features while generating a simplified collision representation of the input model.

IV. Results

A. Coverage of CAD Models

Figure 4e shows a hollow pipe model simplified to an accuracy of 1 mm using the bounding sphere simplification algorithm; the pipe is represented by 419 spheres. Figure 5 shows the bounding sphere simplification algorithm applied to an industrial robot creating a representation of 180 spheres with an accuracy of 20 mm.

The CAD model of a hollow pipe shown in Figure 4a was simplified using sphere packing and the bounding sphere simplification algorithm. Figure 6 shows the maximum error of the bounding sphere simplification and compares it to the maximum error in a sphere packing algorithm as spheres are added. The maximum error in this case is defined as the distance from the point on the mesh surface that lies furthest away from any sphere surface that represents the CAD model. The bounding sphere simplification algorithm used a minimum external sphere size of 100 mm. The maximum error of the sphere packing method is much higher than the maximum error of the bounding sphere simplification algorithm. The graphs are stepped because new spheres are being placed near the point of maximum error. Placing a sphere at the point of maximum error means that there will be two points on either side of the newly inserted sphere with roughly the same error as each other and approximately half of the error that was calculated when the sphere was inserted.

The volume of the of the original CAD model that is covered is also much greater using the bounding sphere

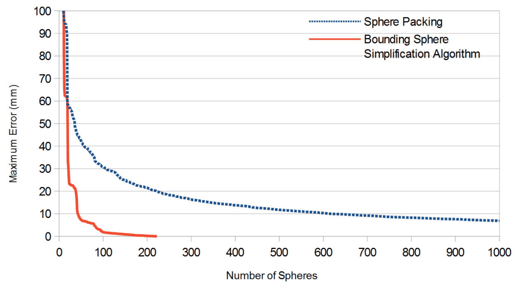

Figure 6: Maximum error of representations as spheres are added.

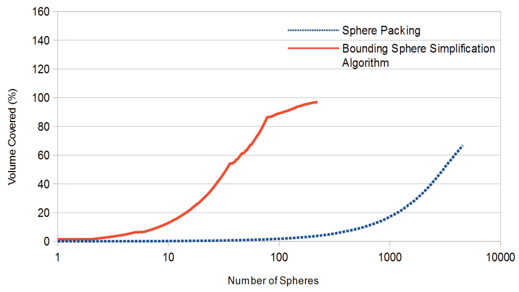

Figure 7: Percentage of the original CAD model covered by spheres.

simplification algorithm. Figure 7 compares the coverage of the two methods as spheres are added to the respective models. The bounding sphere simplification algorithm reaches approximately 97 % coverage of the original CAD model at 222 spheres. The sphere packing method reaches 66 % coverage of the original model at 4546 spheres. At this point the algorithm terminated as the maximum error reached 1 mm. This means that the produced representation contains many gaps smaller than 1 mm between the spheres. The gaps between the spheres can be observed in Figure 4d.

The spheres produced by the bounding sphere simplification cover a significant volume of space outside the bounds of the original CAD model. This volume is the hollow section of the pipe that has been simplified. Figure 8 shows that an additional 145 % of the volume of the pipe is covered by the spheres generated. This means that the spheres produced cover 245 % of the volume of the original model. The sphere packing method covers only an additional 7 % of volume outside the bounds of the original CAD model. The additional coverage increases linearly with the number of spheres for the sphere packing method.

B. Comparison of Collision Detection Performance

The experimental test environment for collision detection performance shown in Figure 9 used an industrial robot to perform simple motions around workspace with a three pipe section that was simplified using different methods. A critical section where the robot moves the Tool Centre Point (TCP) of its weld torch very close to the object, marginally colliding in some cases, has been singled out and the amount of collisions that occurred in this section was recorded to compare the

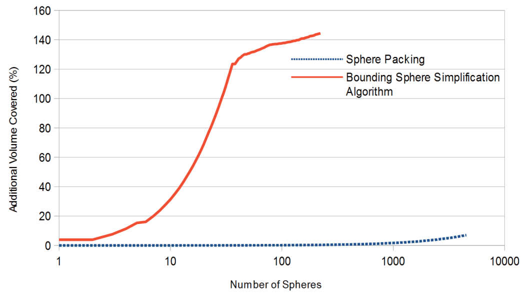

Figure 8: Sphere coverage outside of the original CAD model.

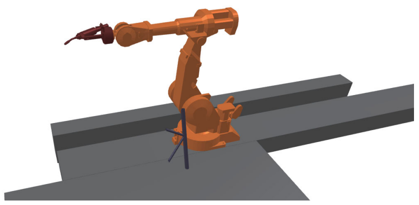

Figure 9: Workspace used for simulation.

relative accuracy of the algorithms. In total 15,178 collision checks were made during each execution of the program.

The results of these simulations are shown in Table I. Both the bounding sphere simplification algorithm and the sphere packing algorithm were run until the maximum error of the simplification was 1 mm. The pipe was also voxelised using a 5 mm grid of 5 mm diameter spheres. The original triangle mesh and a decimated version with approximately half the amount of triangles were used for comparison. The robot, tool and other objects were left as triangle meshes. The torch and robot contain 4318 and 6825 triangles respectively. The program was run again after simplifying the robot, tool and the pipes using the bounding sphere simplification algorithm so that the robot contained 180 spheres and the tool contained 63 spheres. The result of this simulation is shown in the last row of Table I.

The original CAD model appears to have the most accurate collision detection. However, computational time was much slower than the other methods; most of which was spent when the tool was close to the pipes, as checking many triangles in concave meshes against each other is very time consuming.

The decimated CAD showed a noticeable improvement over the original CAD, however there was a loss of accuracy due to the way a decimation algorithm fails to preserve the geometry during the simplification process.

Method	Complexity	Collisions	Time (s)
Original CAD	1942 triangles	2032	1798
Decimated CAD	996 triangles	1801	894
Sphere packing	4752 spheres	2134	258
Voxelisation	2675 spheres	2865	170
Bounding sphere	283 spheres	2222	102
Simplified robot and tool	-	2289	7

Table I: Performance comparison of different model representations.

The sphere based representations all provided a noticeable performance improvement over the triangle mesh representations. The voxel based representation is unusable due to its low accuracy. The sphere packing and bounding sphere representations are both very usable. They both produced a similar level of accuracy to the original CAD model and significantly better performance. However, the bounding sphere simplification algorithm produces a set of spheres significantly smaller than the other sphere based methods and therefore resulted in the best performance.

When the robot and tool were also simplified using the bounding sphere simplification algorithm the runtime was reduced dramatically achieving over 2,000 collision checks per second for the test. There were slightly more collisions detected during the critical section of the test due to the conservative nature of the simplification applied. This did not affect the ability of a motion to be planned using the results.

C. Future Work

Sharp Concave Edges: The bounding sphere simplification algorithm as presented can create inaccurate results when there is a sharp concave edge in a triangle mesh. In this case, the external bounding spheres are not able to be positioned close enough to these edges to provide an accurate representation of the original shape. As a result, the shape is not approximated accurately by the internal spheres. Reducing the minimum external sphere size improves the approximation at these edges but causes the algorithm to not simplify other sections of the mesh. Further work to deal with sharp concave edges is required. Either by fitting other primitives such as boxes or detecting and handling these edges differently.

External Sphere Sets: Ideally every section of the model that could come into contact with an object in the simulation would be represented very accurately and be bound by many small spheres. Sections that are not able to be involved in collisions would not be represented with a high accuracy. In a robotics application, generally an end effector on a robot is the smallest object which will come into contact with any other objects. Instead of creating the bounding spheres individually, it is possible to create them in sets of spheres that represent an object such as an end effector. In this way a further reduction in the number of spheres produced is possible due to more aggressive simplification of sections of the model that will not be involved in collisions.

V. Conclusion

The bounding sphere simplification algorithm presented in this paper produces CAD models that significantly improve performance in collision detection, and as a result AOLP is able to be used in more complex situations. Because of the simplification stage, the bounding sphere simplification algorithm produces models containing fewer spheres than those produced with sphere packing algorithms. The performance increases are most dramatic on CAD models that contain many small features that are removed during the simplification stage of the algorithm. However, at its worst case the bounding sphere simplification algorithm performs no differently than sphere packing algorithms. Because the bounding sphere simplification algorithm is conservative in nature, it is very well suited to generating models that represent real world objects, particularly where collisions must be avoided, as is the case in AOLP. With improved collision detection performance and conservative simplification, a more reliable program can be generated for an industrial robot in less time using AOLP and the bounding sphere simplification algorithm.

VI. Acknowledgements

This work was supported by the Defence Materials Technology Centre (DMTC), which was established and is supported by the Australian Government's Defence Future Capability Technology Centre (DFCTC) initiative.

References

[1] G. Bradshaw and C. O'Sullivan, "Adaptive medial-axis approximation for sphere-tree construction," *ACM Transactions on Graphics*, vol. 23, no. 1, pp. 1–26, 2004.

[2] F. Conti, O. Khatib, and C. Baur, "Interactive rendering of deformable objects based on a filling sphere modeling approach," in *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, vol. 3. IEEE, 2003, pp. 3716–3721.

[3] W. J. Schroeder, J. A. Zarge, and W. E. Lorensen, "Decimation of triangle meshes," in *ACM Siggraph Computer Graphics*, vol. 26, no. 2. ACM, 1992, pp. 65–70.

[4] B. Hamann, "A data reduction scheme for triangulated surfaces," *Computer aided geometric design*, vol. 11, no. 2, pp. 197–214, 1994.

[5] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The quickhull algorithm for convex hulls," *ACM Transactions on Mathematical Software (TOMS)*, vol. 22, no. 4, pp. 469–483, 1996.

[6] J.-M. Lien and N. M. Amato, "Approximate convex decomposition of polygons," in *Proceedings of the twentieth annual symposium on Computational geometry*. ACM, 2004, pp. 17–26.

[7] K. Mamou and F. Ghorbel, "A simple and efficient approach for 3d mesh approximate convex decomposition," in *Image Processing (ICIP), 2009 16th IEEE International Conference on*. IEEE, 2009, pp. 3501–3504.

[8] P. Lindstrom, D. Koller, W. Ribarsky, L. F. Hodges, N. Faust, and G. A. Turner, "Real-time, continuous level of detail rendering of height fields," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM, 1996, pp. 109–118.

[9] G. v. d. Bergen, "A fast and robust gjk implementation for collision detection of convex objects," *Journal of Graphics Tools*, vol. 4, no. 2, pp. 7–25, 1999.

[10] S. W. Wang and A. E. Kaufman, "Volume sampled voxelization of geometric primitives," in *Visualization, 1993. Visualization'93, Proceedings., IEEE Conference on*. IEEE, 1993, pp. 78–84.

[11] D. Cohen-Or and A. Kaufman, "3d line voxelization and connectivity control," *Computer Graphics and Applications, IEEE*, vol. 17, no. 6, pp. 80–87, 1997.

[12] X. Jia and R. Williams, "A packing algorithm for particles of arbitrary shapes," *Powder Technology*, vol. 120, no. 3, pp. 175–186, 2001.

[13] K. Han, Y. Feng, and D. Owen, "Sphere packing with a geometric based compression algorithm," *Powder Technology*, vol. 155, no. 1, pp. 33–41, 2005.

[14] J.-F. Jerier, D. Imbault, F.-V. Donze, and P. Doremus, "A geometric algorithm based on tetrahedral meshes to generate a dense polydisperse sphere packing," *Granular Matter*, vol. 11, no. 1, pp. 43–52, 2009.

[15] P. M. Hubbard, "Approximating polyhedra with spheres for time-critical collision detection," *ACM Transactions on Graphics (TOG)*, vol. 15, no. 3, pp. 179–210, 1996.

[16] C. Andújar, P. Brunet, and D. Ayala, "Topology-reducing surface simplification using a discrete solid representation," *ACM Transactions on Graphics (TOG)*, vol. 21, no. 2, pp. 88–105, 2002.