TUM

TECHNISCHE UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK

The Bounding Mesh Algorithm

Andre Gaschler, Quirin Fischer, Alois Knoll

TUM-I1522

The Bounding Mesh Algorithm

Andre Gaschler, Quirin Fischer and Alois Knoll

June 9, 2015

Abstract

We present an algorithm to generate a *one-sided* approximation of a given triangular mesh. We refer to such an approximate mesh as a *bounding mesh*, which includes the original mesh and has fewer vertices. Likewise, an inner bounding mesh is defined as an approximate mesh that is included by a given mesh. Our proposed bounding mesh algorithm performs iterative edge contractions and can generate both types of approximation.

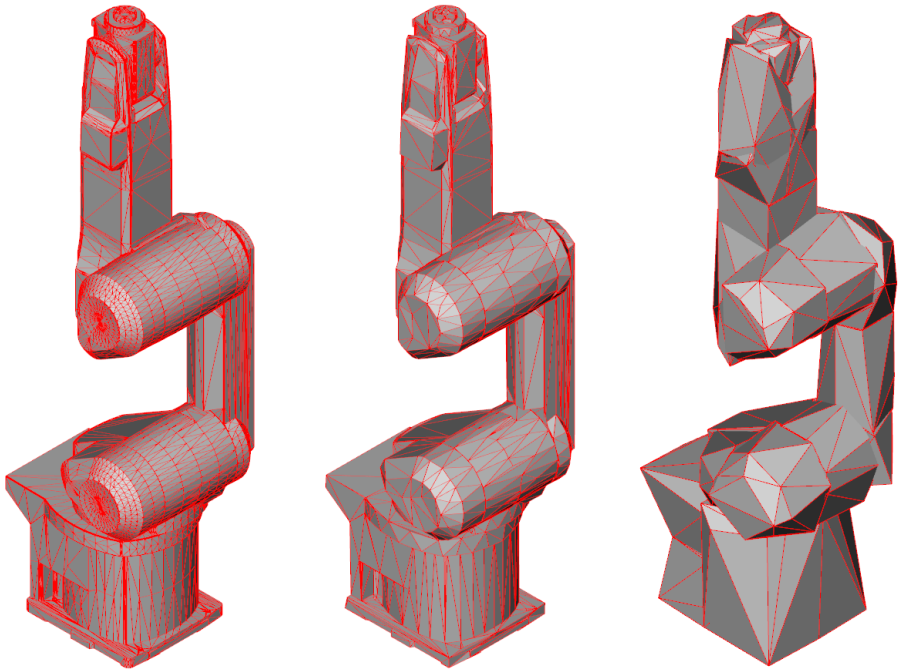
Contrary to regular, two-sided mesh approximation, which is a well studied subject in computer graphics, our algorithm is novel and one of a handful approaches to one-sided mesh approximation. While we are the first to apply bounding meshes to safe collision detection, path planning, and robot motion planning, applications range further to computer geometry and computer graphics. The bounding mesh algorithm helps pre-processing complex geometries and increases the efficiency of existing geometric algorithms, especially those that search in a bounding volume hierarchy. It can speed up search, intersection and inclusion detection, as well as silhouette, clipping, and other operations, acting as an intermediate level of approximation between coarser bounding boxes or bounding spheres and the exact mesh.

Furthermore, the bounding mesh algorithm combines well with approximate convex decomposition to generate a bounding set of convexes with very few vertices, which is an efficient data structure for intersection, distance and normal computation, as well as other geometric operations.

1 Introduction

Most computer geometry algorithms operate more efficiently when bounding volume hierarchies of a given mesh are available. Bounding boxes, bounding spheres, convex hulls, axis-aligned boxes, or discrete orientation polytopes have long been used to simplify broad-phase queries in collision checking [8, 13] and other problems. While these approximations are potentially far from the original mesh, a bounding mesh (Fig. 1b) offers a fine approximation whose distance to the original mesh can be controlled. Importantly, the approximation is one-sided (or, single-sided), which is necessary for applications such as safe collision and inclusion detection, upper and lower distance approximation, motion planning, and many geometric algorithms that operate on level-of-detail models.

Bounding meshes can rather easily be generated by iterative edge contraction with additional constraints to achieve a local one-sided approximation. In



(a) $V = 9296$ (b) $V = 2221$ (c) $V = 220$

Figure 1: Complex meshes (a) are simplified effectively by our new bounding mesh (b) and bounding sets convex polyhedra (c) algorithms.

addition, the bounding mesh algorithm integrates well with approximate convex segmentation [12] to generate bounding sets of convex polyhedra for a given mesh (Fig. 1c). A bounding set of convex polyhedra has fewer vertices than a convex decomposition and therefore increases the efficiency of many geometric algorithms. As an example, bounding sets of convexes allow safe and efficient robot motion planning. Since collision and distance queries require only convex–convex queries with few vertices, computation can even be guaranteed within real-time limits.

While we first sketched the idea of bounded meshes and their convex decomposition in an application to robot task and motion planning [6, 4, 5], this paper gives a complete derivation of our bounded mesh algorithm and evaluates its effectiveness. In the following, we first describe our work with respect to related geometric algorithms and applications in Section 2. Then, we derive our bounding mesh algorithm as a constrained approximation problem in Section 3. Finally, we evaluate our approach in Section 4 with respect to the quality of the approximation.

2 Related Work

The idea to generate simpler meshes that enclose a given mesh was probably first described by Hoppe in a 1999 patent description [10]. Hoppe describes a progressive hull structure suitable for progressive transmission and rendering,

which is generated by a sequence of edge contractions. Unlike our definition of edge contractions, which minimize a distance metric, Hoppe's progressive hulls minimize the added volume, which often leads to sharp spikes, possibly far away from the original mesh. While Hoppe's metric is well suited for graphics algorithms, ours is optimized towards collision, inclusion, and distance queries.

Gu et al. [9, pp. 21–23] perform a one-sided approximation in both directions to speed up an algorithm for silhouette clipping. Sander et al. [16] later extend this work to approximate texture and normal mapping.

Platis and Theoharis [15, 14] borrow Sander's one-sided mesh approximation to increase the efficiency their algorithm for ray–mesh intersection points. Cholt [1] later proposes a few technical improvements.

Ciesla [2] describes a purely geometric approach to one-sided mesh approximation, which does not minimize a specific cost.

According to an extensive literature survey, the above publications are the only prior works in one-sided mesh approximation. Note that almost all minimize the added volume [10, 9, 16, 15, 14, 1] or have no specific cost function [2]. Our approach [4, 5], however, minimizes a well-defined distance measure, resulting in bounding meshes at very low Hausdorff distances from the input mesh, which is highly desirable for approximate intersection, inclusion, and distance queries.

3 Approach

Our bounding mesh algorithm performs iterative edge contractions to simplify a given mesh in a single direction to construct an outer or an inner bounding mesh. Contrary to earlier works, we derive an error metric that approximates the Hausdorff distance.

3.1 Bounding Mesh Generation

Similar to the two-sided mesh approximation algorithm by Garland and Heckbert [3], we perform iterative edge contraction steps that are guided by a quadric error measure. Constraining the selection of the contraction point to lie above all neighboring triangles, we can guarantee for the resulting mesh to enclose the original mesh; it becomes a bounding mesh. When the constraints are negated, an inner bounding mesh is generated. Note that the existing quadric error measure [3] underestimates distances in the case of acute angles. To guarantee the Hausdorff distance as a lower bound, we add an additional term to the error measure.

3.2 Edge Contraction Cost Function

The most important choice in our bounding mesh algorithm is the design of the edge contraction cost function $E : (e, \boldsymbol{v}) \mapsto \mathbb{R}$, where e is an edge of points $(\boldsymbol{v}_1, \boldsymbol{v}_2)$ that are contracted to single a vertex $\boldsymbol{v}$, as depicted in Fig. 2. The goal of a bounded edge contraction is to find a contraction point $\boldsymbol{v}^*$ that is restricted outside (or, inside) the mesh and has minimal cost $E(e, \boldsymbol{v}^*)$. Denoting P as the neighboring planes of a vertex, we can formulate the search for the contraction

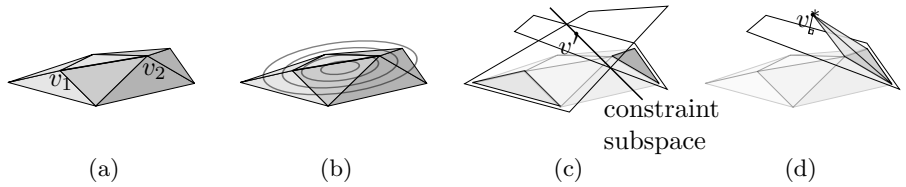


Figure 2: Bounded edge contraction. (a) Edge with neighboring planes, (b) quadric cost matrix $\boldsymbol{Q}$, (c) example of a case $m = 2$ minimizer, (d) global constraint minimizer v^* adds tetrahedron volumes on each plane that have positive heights

point $\boldsymbol{v}^*$ as a constrained minimization problem

$$\min_{\boldsymbol{v}^*} \; E((\boldsymbol{v}_1, \boldsymbol{v}_2), \boldsymbol{v}^*) \;\; \text{s.t.} \;\; \forall \boldsymbol{p} \in P(\boldsymbol{v}_1) \cup P(\boldsymbol{v}_2) : \;\; \boldsymbol{p}^{\mathrm{T}} \boldsymbol{v}^* \geq 0 \,. \tag{1}$$

In order to generate an inner bounding mesh, the inequality in Eq. 1 is negated to $\boldsymbol{p}^{\mathrm{T}} \boldsymbol{v}^* \leq 0$. The only previous bounding mesh approaches [16, 15, 10] suggest to define E as the added volume, as shown in Fig. 2d. While this may be a reasonable choice for some graphics applications, the added volume may become infinitely small in common cases such as sharp angles or thin triangles, effectively allowing $\boldsymbol{v}^*$ to be arbitrarily far away; this would create bounding meshes unacceptable for distance-measuring, collision-checking, or other applications.

Therefore, our goal is to define a cost function E that is rather close to the real Hausdorff distance. Luebke [11] discusses several such cost functions, of which we choose an adaptation of the quadric error measure proposed by [3]. Let d denote the Hausdorff distance of two geometric entities. Intuitively, the Hausdorff distance between two meshes is defined as the maximum distance an arbitrary point on one mesh can have with its closest point on another. The error quadric approximates the real distance of a point $\boldsymbol{v}$ to the mesh by the sum of squared distances to the neighboring planes $P(e)$; this can conveniently be written as a multiplication with a symmetric 4-by-4 matrix $\boldsymbol{Q}(e)$.

$$E(e, \boldsymbol{v}) = \sum_{\boldsymbol{p} \in P(e)} \left(d(\boldsymbol{p}, \boldsymbol{v})\right)^2 = \boldsymbol{v}^{\mathrm{T}} \sum_{\boldsymbol{p} \in P(e)} \boldsymbol{p}^{\mathrm{T}} \boldsymbol{p} \; \boldsymbol{v} = \boldsymbol{v}^{\mathrm{T}} \boldsymbol{Q}(e) \; \boldsymbol{v} \tag{2}$$

Fig. 2b visualizes this type of quadric cost function for a placement of $\boldsymbol{v}$. The matrix $\boldsymbol{Q}$ above is the one proposed by [3] and is frequently used for its simplicity [11]. One important benefit of $\boldsymbol{Q}$ is that after an edge collapse, the cost functions of modified edges do not need to be re-calculated as a sum of $\boldsymbol{p}^{\mathrm{T}} \boldsymbol{p}$, but can simply be approximated by the upper bound

$$\boldsymbol{Q}(\boldsymbol{v}^*) = \boldsymbol{Q}(\boldsymbol{v}_1) + \boldsymbol{Q}(\boldsymbol{v}_2). \tag{3}$$

This summation step can therefore be calculated in constant time rather than a time linear to the number of vertices.

However, there are two issues that should be resolved for this metric to approximate the Hausdorff distance more closely:

1. Distances to planes can underestimate the distance to triangles. Because of this, Eq. 2 may be infinitely smaller than the real distance to the mesh,

```
for every edge e do
    Compute Q(e) (Eq. 4);
    Minimize E(e, v*) (Eq. 1);
    Add E(e, v*) to priority queue;
end
while E(e, v*) < ε² do
    Pop best edge e;
    Modify geometry
        Remove edge e and adjacent triangles;
        Insert new vertex v* and triangles;

    Modify priority queue
        Re-calculate costs of changed edges;

end
```

Algorithm 1: Bounding Mesh Generation with iterative edge contraction

even when we assume $\boldsymbol{v}$ to be in the constrained subspace defined by Eq. 1. Intuitively, this happens in the very common case of acute angles. In the following section, we propose a new definition of $\boldsymbol{Q}$ to fix this issue.

2. While the approximated cost update in Eq. 3 is an upper bound, it may be three times as high as the real cost and cause sub-optimal results [3]. We fix this by appropriate subtractions using the inclusion-exclusion principle, which can also be applied to earlier works [4, 3].

3.3 Quadric Hausdorff Distance Approximation

Let us consider the case of approximating the Hausdorff distance between a point $\boldsymbol{x}$ and an edge with its two neighboring planes $\{e, \boldsymbol{p}_1, \boldsymbol{p}_2\}$. We can argue that infinitesimally beveling or rounding an edge does not change its Hausdorff metric. Therefore, a possible choice for an additive term to the edge cost $\boldsymbol{Q}(e)$ can be the distance to a mean or "bevel" plane normal to the edge e. Mathematically, we define a corrected cost matrix for an edge $\boldsymbol{Q}(e)$ with neighboring planes $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ with the added normal plane:

$$\boldsymbol{Q}(e) := \boldsymbol{p}_1^{\mathrm{T}}\boldsymbol{p}_1 + \boldsymbol{p}_2^{\mathrm{T}}\boldsymbol{p}_2 + \alpha\ (\boldsymbol{p}_1 + \boldsymbol{p}_2)^{\mathrm{T}}(\boldsymbol{p}_1 + \boldsymbol{p}_2)\,. \tag{4}$$

We want to ensure it is not smaller than the Hausdorff metric, i.e., $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}(e)\,\boldsymbol{x} \geq d(\{e, \boldsymbol{p}_1, \boldsymbol{p}_2\}, \boldsymbol{x})^2$ holds for all points $\boldsymbol{x}$. It can be shown that the minimal α fulfilling this equation is $\alpha = \langle \boldsymbol{n}_1, \boldsymbol{n}_2 \rangle$, with $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ being the normals of the planes $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$.

3.4 Iterative Edge Contraction

Now that we have defined a feasible Hausdorff distance approximation that can be updated when contracting edges, we devise an algorithm to minimize edge contraction costs and to finally generate the bounding mesh. Our overall bounding mesh generation procedure is outlined in Algorithm 1. As proposed by

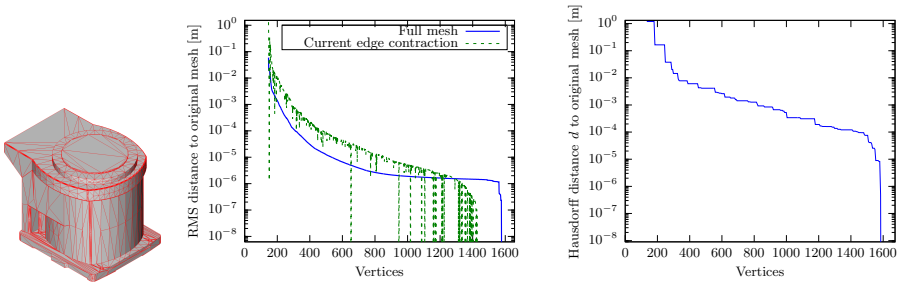
Garland and Heckbert [3], costs to collapse an edge can efficiently be stored in a priority queue. When an edge is contracted, only cost functions of neighboring edges need to be recalculated and moved in the queue.

In the initialization step, the algorithm starts computing costs for the collapsing of all edges, and inserts them into the priority queue. Minimizing the contraction cost of an edge e from Eq. 1 is a quadratic problem with linear inequalities. However, as visualized in Fig. 2c, it is evident that the minimizer $\boldsymbol{v}^*$ may be constrained by zero, one, two, or at least three planes. We can therefore exhaustively search for all candidate minimizers in all four cases $m \in [0..3]$ by enumerating all m-subsets of neighboring planes $P(\boldsymbol{v}_1) \cup P(\boldsymbol{v}_2)$ [4]. It is straightforward to construct the $(3-m)$-dimensional subspace of each constraint subset; Fig. 2c shows an $m = 2$ case with the intersection of the two planes of neighboring triangles. We can then solve for each candidate minimizer $\boldsymbol{v}'$ by unconstrained quadratic minimization in the respective subspace. With this approach, all four cases $m \in [0..3]$ reduce to linear systems with up to three dimensions. Of all candidate minimizers that fulfill all inequality constraints, we compute the cost function and finally obtain the global minimizer $\boldsymbol{v}^*$. Note that since the number of neighboring planes is very limited, our exhaustive search is sufficiently fast, with 10^5 edge decimations per second on an average desktop computer [4].

3.5 Bounding Convex Decomposition

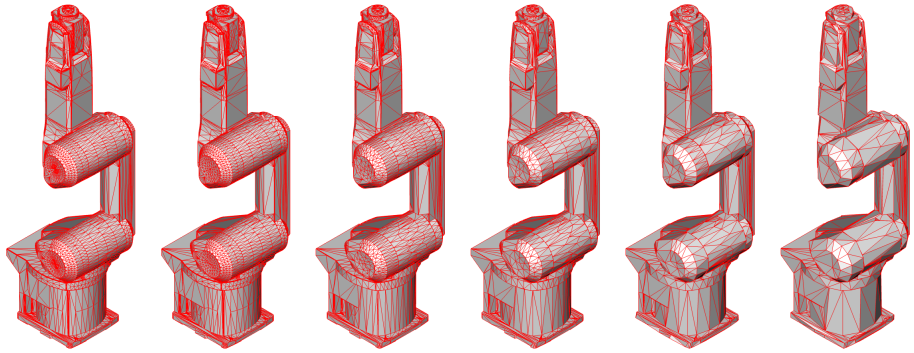
Up to this step, we have simplified a given mesh to a bounding mesh, as shown in Fig. 1b and 4. Even though this bounding mesh is a precise approximation, it is non-convex and rather inefficient to process for distance-measuring and many motion planning algorithms. The crucial step is to decompose it into convex polyhedra and obtain a bounded set of convexes, as shown in Fig. 1c. To achieve this bounded convex decomposition, we adapt the approximate convex segmentation by Mamou and Ghorbel [12]. Their algorithm searches on the dual graph of triangles to contract edges and to segment convex clusters. The segmentation is guided by a weighted cost function of a concavity and an aspect ratio measure. The aspect ratio cost is designed to dominate the first few iterations and to quickly simplify the dual graph; after that, the segmentation is mostly lead by the concavity measure.

Note that the convex segmentation itself does not reduce the number of vertices of the mesh. Mamou and Ghorbel therefore use (two-sided) mesh simplification [3] in a preprocessing step of their implementation [12], and further discard all but a constant number (i.e., 16) of vertices of each convex cluster. Importantly, both of these steps do not preserve the volume of the original mesh, and do not generate a bounding set of convex polyhedra [7]. We therefore replace both steps by our bounding mesh algorithm. Note that convex decomposition alone would hardly reduce the complexity of meshes (as it leaves convex areas unchanged), so only the combination of our work and [12] can generate bounding sets of convex polyhedra. These bounding sets of convexes precisely approximate the real geometry (see Fig. 1c) and, being convex, can even guarantee worst case collision detection within real-time limits.



(a) (b) (c)

Figure 3: ROBOT BASE: Bounding mesh precision with respect to vertex count. (a) Original mesh, (b) quadric cost function E with respect to vertex count, (c) exact Hausdorff distance d to original mesh



$V = 9296$ (original)	$V = 8434$ $\epsilon < 0.0001\,\mathrm{m}$	$V = 5382$ $\epsilon < 0.001\,\mathrm{m}$	$V = 3967$ $\epsilon < 0.002\,\mathrm{m}$	$V = 3015$ $\epsilon < 0.005\,\mathrm{m}$	$V = 2280$ $\epsilon < 0.017\,\mathrm{m}$

Figure 4: A series of bounding meshes tightly approximates a ROBOT geometry by enclosing meshes, greatly reducing the number of vertices.

4 Evaluation

To evaluate the performance of our approach, we analyze the approximation quality of our bounding mesh algorithm.

With our approach, a typical robot geometry mesh can easily be simplified to a bounding mesh using only a fraction of the number of vertices V, as shown in Fig. 4. In the example, the bounding meshes have acceptable Hausdorff distances ϵ of a few millimeters at one third of the number of vertices; high-definition CAD models may be simplified even further. As a result, collision detection will safely recognize all collisions, and can possibly report false positives when distances are within $[0, \epsilon]$. In contrast to standard, two-sided mesh simplification [3, 12], it will never fail to detect a collision.

Fig. 3 gives a more detailed view on the approximation error, with respect to the number of collapsed edges. As shown in Fig. 3b, the mesh of a typical robot rigid body (the base part of the one shown in Fig. 4), with initially 1590 vertices, will easily reduce to a bounding mesh of 800–1000 vertices at very little cost. At 300–400 vertices, there is a "sweet spot" with an acceptable Hausdorff

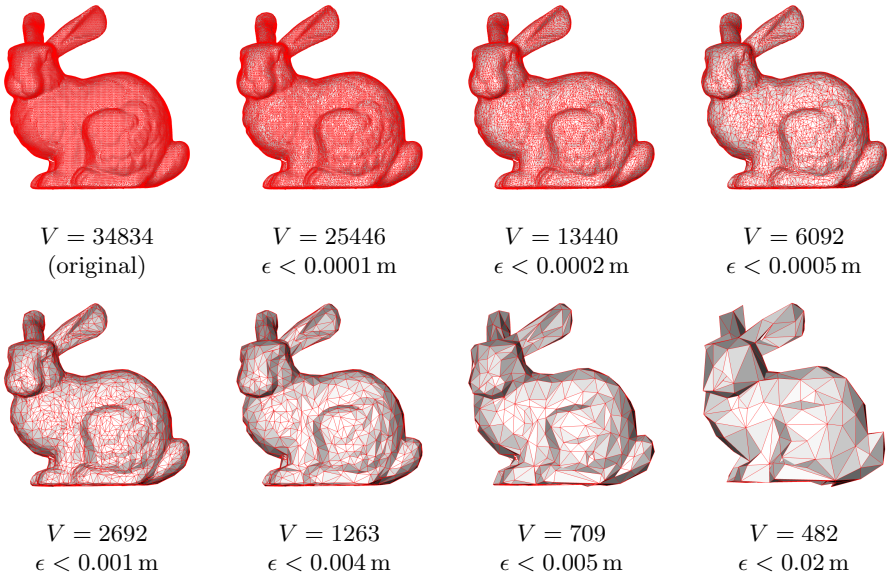


Figure 5: A series of bounding meshes of the common benchmark geometry Stanford Bunny with decreasing numbers of vertices.

distance and good reduction; with even lower vertex counts, the approximation error then steeply increases. In the same graph, the dashed line shows marginal costs of single edge collapses; notably, some of them even reduce the approximation error, which is indicated by lower peaks (negative scale is not shown). Fig. 3c shows the same bounding mesh simplification, but with exact Hausdorff distances calculated. Because our cost function E is based on a sum of squared distances, Hausdorff distances are at the order of $\sqrt{E}$. This graph indicates that our cost function is a viable approximation of the Hausdorff distance to generate ϵ-precise bounding meshes.

For comparison, we also included the well-known Stanford Bunny with its bounding meshes in Fig. 5. Because this mesh was triangulated rather uniformly at great detail, our bounding mesh generation performs particularly well at reducing the numbers of vertices at very low Hausdorff distances ϵ.

5 Conclusion and Future Work

In this work, we propose an algorithm for approximating arbitrary meshes by bounding meshes, which enclose the original mesh and have fewer vertices. Bounding meshes may be applied to a wide range of geometric algorithms, including fast and safe collision checking, and lower and upper approximate distance computation. In combination with convex decomposition, we can generate bounding sets of convexes, a data structure that allows very efficient geometric algorithms, and even collision detection within guaranteed worst case time limits.

Future work will include an open source release of the software discussed,

made available on the author's website[1], and more research on inner bounding meshes and their application to inclusion queries.

Acknowledgements

This research was supported by the European Union's Seventh Framework Programme through the projects JAMES under grant agreement no. 270435 [2] and SMErobotics under grant agreement no. 287787 [3].

References

[1] David Cholt. Progressive Hulls: Application on Biomedical Data. In *Central European Seminar on Computer Graphics for students*, 2012.

[2] Christoph Roland Ciesla. Development of a system for the reduction of 3d polygon meshes in the field of robotics. Diplomarbeit, Technische Universität München, 2007.

[3] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the International Conference on Computer Graphics and Interactive Techniques*, pages 209–216, 1997.

[4] A. Gaschler, R. P. A. Petrick, O. Khatib, and A. Knoll. A knowledge of volumes approach to robot task planning. 2015. submitted.

[5] Andre Gaschler, Ingmar Kessler, Ronald P. A. Petrick, and Alois Knoll. Extending the knowledge of volumes approach to robot task planning with efficient geometric predicates. In *IEEE International Conference on Robotics and Automation (ICRA)*, June 2015.

[6] Andre Gaschler, Ronald P. A. Petrick, Manuel Giuliani, Markus Rickert, and Alois Knoll. KVP: A knowledge of volumes approach to robot task planning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 202–208, 2013.

[7] Andre Gaschler, Ronald P. A. Petrick, Torsten Kröger, Oussama Khatib, and Alois Knoll. Robot Task and Motion Planning with Sets of Convex Polyhedra. In *Robotics: Science and Systems (RSS) Workshop on Combined Robot Motion Planning and AI Planning for Practical Applications*, 2013.

[8] Elmer G. Gilbert, Daniel W. Johnson, and S. Sathiya Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal of Robotics and Automation*, 4(2):193–203, 1988.

[9] Xianfeng Gu, Steven J Gortler, Hugues Hoppe, Leonard McMillan, Benedict Brown, and Abraham Stone. Silhouette mapping. *Computer Science Technical Report TR-1-99, Harvard University*, 1999.

[1] `http://www.boundingmesh.com/`
[2] `http://www.james-project.eu/`
[3] `http://www.smerobotics.org/`

[10] Hugues H Hoppe. Progressive hulls, July 1 2003. US Patent 6,587,104.

[11] David P Luebke. *Level of Detail for 3D Graphics*. Morgan Kaufmann Pub, 2003.

[12] Khaled Mamou and Faouzi Ghorbel. A simple and efficient approach for 3D mesh approximate convex decomposition. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, pages 3501–3504, 2009.

[13] Jia Pan, Sachin Chitta, and Dinesh Manocha. FCL: A general purpose library for collision and proximity queries. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3859–3866, 2012.

[14] Nikolaos V. Platis. *Τεχνικές πολλαπλών αναλύσεων στην απλοποίηση τριγωνικών και τετραεδρικών πλεγμάτων (Multiresolution Techniques for the Simplification of Triangular and Tetrahedral Meshes)*. PhD thesis, University of Athens, 2005.

[15] Nikos Platis and Theoharis Theoharis. Progressive hulls for intersection applications. In *Computer Graphics Forum*, volume 22, pages 107–116, 2003.

[16] Pedro V Sander, Xianfeng Gu, Steven J Gortler, Hugues Hoppe, and John Snyder. Silhouette clipping. In *Proceedings of the International Conference on Computer Graphics and Interactive Techniques*, pages 327–334, 2000.