Generating Approximative Minimum Length Paths in 3D for UAVs

Flemming Schøler, Anders la Cour-Harbo, and Morten Bisgaard

Aalborg University, Fredrik Bajers Vej 7C, Aalborg University, 9220 Aalborg East, Denmark.

E-mail: {fls, alc, bisgaard}@es.aau.dk.

Abstract

We consider the challenge of planning a minimum length path from an initial position to a desired position for a rotorcraft. The path is found in a 3-dimensional Euclidean space containing a geometric obstacle. We base our approach on visibility graphs which have been used extensively for path planning in 2-dimensional Euclidean space. Generalizing to 3-dimensional space is not straight-forward, unless a visibility graph is generated that, when searched, will only provide an approximative minimum length path. Our approach generates such a visibility graph that is composed by an obstacle graph and two supporting graphs. The obstacle graph is generated by approximating a mesh around the configuration space obstacle, which is build from the convex hull of its work space counterpart. The supporting graphs are generated by finding the supporting lines between the initial or desired position and the mesh. An approximation to the optimal path can subsequently be found using an existing graph search algorithm. The presented approach is suitable for fully known environments with a single truly 3-dimensional (not merely "raised" 2-dimensional) obstacle. A example for generating a path for a small-scale helicopter operating near a building is shown.

keywords: UAV, path planning, helicopter, visibility graph

1 INTRODUCTION

Path planning and trajectory generation are fundamental areas for UAS development and both provide the ability to figure out a way to efficiently and safely travel through an environment under a set of constraints. Specifically, for tasks such as surveillance, inspection, aerial mapping, etc., small-scale autonomous helicopters are increasingly being used. In many such tasks it is advantageous to operate in close proximity to the obstacles or to follow their surface.

1.1 Background

The practical use of the presented work is to obtain methods for path planning that can be used for operating a small-scale helicopter in an environment constrained by obstacles. The presented method is near-optimal, that is, able to produce a path arbitrarily close to the Euclidean shortest path inside a space. This space is constrained by a single obstacle such that no parts of the vehicle may intersect the obstacle at any time. In this space the vehicle is represented by its bounding sphere, which is centered in center-of-mass. The presented approach for path planning use visibility graphs (VG). Theses VGs are based on describing the obstacles in a configuration space (CS), which in turn are based on the work space (WS). The generation of those CS obstacles and VGs are the main contribution of this paper.

The WS obstacle can be build from any set of vertices such as a point-cloud from sampled data. The WS obstacle is made convex since non-convex parts are not relevant when searching the VG for a solution. In fact, when computing the shortest path between two points in the CS, the shortest path is a series of connected edges that are either on the convex hull of the obstacle, or tangents from the points to the obstacle.

1.2 Previous work

Previous work on how to practically, automatically, and efficiently generate a configuration space from a 3D point-cloud in WS seems limited to building primitive solids or geometric representations, and in many path planning publications an existing configuration space is simply assumed.

Visibility graphs for 2D path planning have been used extensively. However finding an optimal solution to the general path planning problem in 3D is NP-complete, see [1, 2]. The difficulty of the problem is that the shortest path around a polyhedral obstacle does not in general traverse only vertices of the polyhedron, but also points somewhere on the edges of the polyhedron. The concept of adding additional vertices along these edges in configuration space so that no edge vertices are spaced more than a specified maximum length is introduced by [1]. This approach generally results in a good approximation to the optimal path.

In [3] an algorithm is presented for the 2D case that use the Minkowski sum (see [4] for details on Minkowski sum) to generate a CS obstacle from a polygonal obstacle and a polygonal vehicle. While CS obstacles could be generated by approximating the vehicle with a geodesic dome, the obstacles are described by a sequence of algebraic equations, which makes the optimization problem difficult to solve without resorting to numerical methods. Also [5] describe algebraic algorithms for generating the boundary of CS obstacles.

1.3 Present work

We propose a method for generating a VG from 1) a point-cloud that represent the WS obstacle, 2) a vehicle bounding sphere with radius r_{bs}, and 3) an initial and desired position for the vehicle. The generalized visibility graph VG(N, L) contains a node set N and a link set L of links between node pairs. Each node in the graph is assigned a possible configuration (or position), and each link represents an visible (linear) connection between them. The VG is composed of two parts; an obstacle visibility graph VG$_o$ that links node-pairs near the obstacle surface, and a supporting visibility graph VG$_{vp}$ that connects the initial and desired position (denoted "via points") to the CS obstacle by supporting lines:

$$\mathrm{VG}_f = \mathrm{VG}_o \cup \mathrm{VG}_{vp} \tag{1}$$

An example of such a graph is shown in Figure 1. The advantage of separating the obstacle and via point

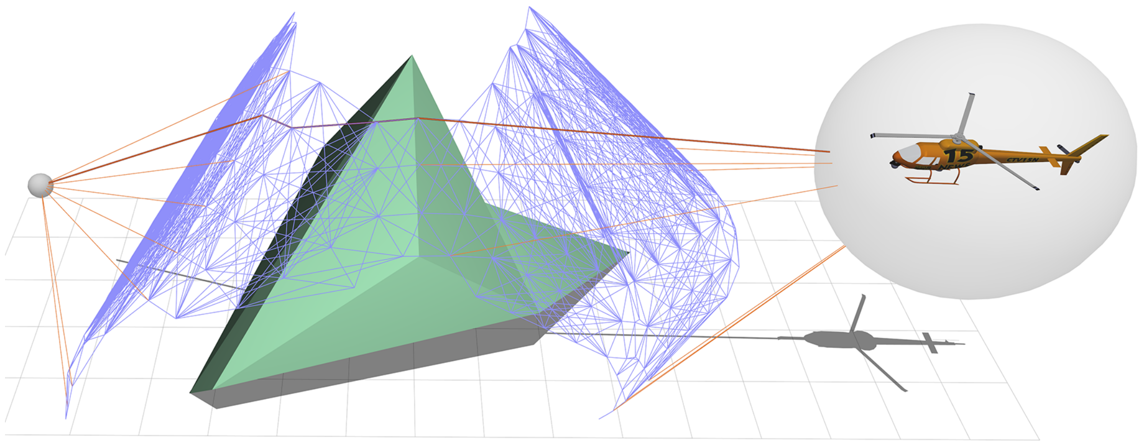


Figure 1: The VG_f for a helicopter flying between two via points around an obstacles. Part of the obstacle visibility graph VG_o is drawn in blue. Part of the supporting graph VG_{vp} is drawn in orange. The shortest found path is red and connects the two via points drawn as gray spheres. The larger sphere represents the vehicle bounding sphere. The CS obstacle is not shown.

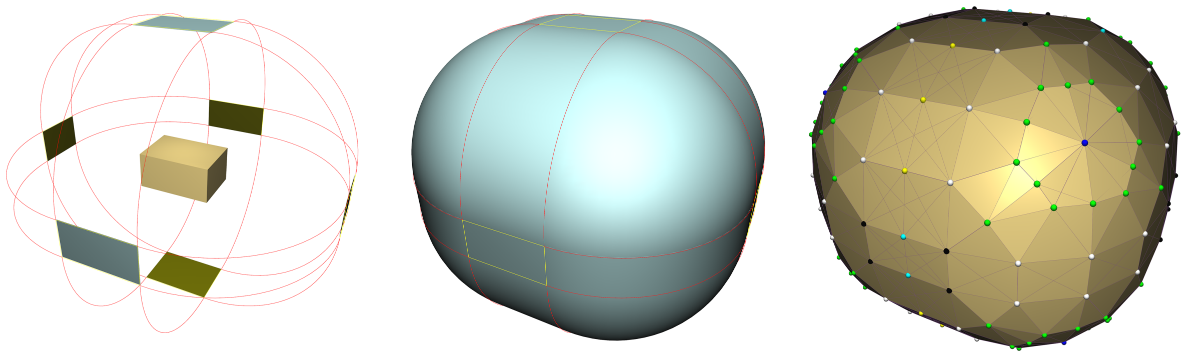
(VP) graphs is that while both graphs must be generated when creating the initial full VG, only VG_{vp} has to be created when either the obstacle is moved or rotated or either via point is moved. The full VG is generated such that it contains only links that are relevant when computing the Euclidean shortest path between via points, and subsequent pruning is not necessarily. The full VG can be searched to obtain the shortest path using a graph search algorithm, e.g. Dijkstra's algorithm, see [6] or A*, see [7]. This part is not the focus of this research and will not be discussed further.

2 METHOD

Generating the obstacle graph VG_o and supporting graph VG_{vp} are treated in Section 2.1 and Section 2.2, respectively.

2.1 Obstacle Visibility Graph

An obstacle visibility graph VG_o is based on the configuration space of a spherical vehicle moving amongst convex, polyhedral obstacles, i.e. obstacles described by facets. Thus, CS obstacles are grown as the Minkowski sum of a sphere and convex WS polyhedra generated from the convex hull (see e.g. [8] for a robust $O(n^2)$ algorithm for finding the convex hull) of the point-cloud. As seen from the example in Figure 2, this is equivalent to generating the CS obstacle by translating each facet of the WS obstacle along its outward pointing normal by r_{bs}, while filling the occurring gaps between edges of translated facets by cylindrical patches, and gaps at facet vertices by spherical patches. Any such generated CS obstacle will have a surface with G1 continuity. As shown in Figure 2(c) the VG of a CS obstacle can be seen as a mesh wrapped around (and fully enclosing) that obstacle. The mesh points acts as nodes in the graph, and the links are edges of a convex hull of the obstacle. The mask size of the mesh is given by the desired accuracy of the near-optimal path. To fully enclose the obstacle, these nodes must be located slightly above the obstacle surface. This distance is denoted r_e and increases as the desired accuracy decreases.



(a) Cuboid in workspace and translated facets

(b) Cuboid in configuration space

(c) Cuboid visibility graph

Figure 2: The configuration space and visibility graph generated from a cuboid. VG nodes are colored according to origin.

The nodes (mesh points) are a combination of the points from the edges of the patches and points internally to the patches. Only cylindrical and spherical patches are used when generating the nodes. Points in facets can be skipped since interior points will not be part of an optimal solution, and edges are shared with cylindrical patches. Once the node set is generated for a patch, the link set can be determined using the convex property of the CS obstacle. Convexity means that links exist only between nodes of the same patch. In the following, we describe how to find the nodes for both patch types.

2.1.1 Sphere patch vertices

The vertices on a sphere patch has four separate origins as seen by the black, white, green, and blue vertices in Figure 2(c). The black endpoint vertices form the sphere patch polygon, the white vertices lie along the polygon edges, and green or blue vertices are inside the polygon. The black vertices are given by the neighboring translated facets, while the other types are interpolated to achieve a sufficiently high resolution of the VG. The blue central vertex is used when interpolating the green vertices.

To obtain the interpolated nodes, let N_{sn} be the number of neighboring cylindrical patches, then the spherical polygon is composed by N_{sn} edges between N_{sn} vertices. The set of these vertices is denoted $\{v_i\}$ for $i = 0, \ldots, N_{\text{sn}} - 1$. Using a central vertex inside the polygon, the spherical patch polygon can now be decomposed into N_{sn} spherical triangles. Each triangle use a different edge but all share the central vertex.

A central vertex that minimize spherical distance to the vertex set is given as

$$\underset{v_c}{\operatorname{argmin}} \sum_{i=0}^{N-1} w_i \mathrm{dist}_{\mathrm{geo}}(v_c, v_i)^2 \ ,$$

where $\mathrm{dist}_{\mathrm{geo}}(v_c, v_i)$ is the geodesic distance from v_c to v_i, and w_i are non-negative weights that sum to 1. This problem is well-defined for vertices on a hemisphere and a fast iterative algorithm with quadratic convergence rate exists, see [9].

All remaining nodes in the patch are found by two consecutive interpolations. The first interpolation is along the edges of the sphere patch polygon to find a set of vertices v_a. The second interpolation is along the edges formed by each of these vertices and the central vertex.

The number of vertices generated by interpolation is controlled by the parameter $l_{\max}$ that determines the maximum allowed surface distance between two neighboring vertices. Since the VG must fully enclose the obstacle, a lower interpolation resolution requires a larger patch radius to ensure no links intersect the obstacle. Both factors affect the near-optimality of solution, since a smaller obstacle and more nodes are more likely to result in a solution closer to optimal.

The number of nodes $N_{\mathrm{se}}(i)$ along an exterior edge i of a spherical polygon is

$$N_{\mathrm{se}}(i) = \lceil (r_{bs} + r_e) \arccos(v_i \cdot v_{i+1}) / l_{\max} \rceil \ ,$$

where $\{v_i\}$ is the set of consecutive endpoint vertices in the spherical polygon. Using the following function for spherical linear interpolation

$$\begin{aligned} p(v, w, s, K) = \Big(1 - (v \cdot w)^2\Big)^{-\frac{1}{2}} \Big(&\sin\Big(\frac{s}{K} \arccos(v \cdot w)\Big)\, w \\ &+ \sin\Big(\Big(1 - \frac{s}{K}\Big) \arccos(v \cdot w)\Big)\, v\Big) \ , \end{aligned}$$

the vertices on the arc between v_i and v_{i+1} are given by

$$v_a(i, s) = p\big(v_i, v_{i+1}, s, N_{\mathrm{se}}(i)\big) \tag{2}$$

and then (with a slight abuse of notation)

$$V_a = \bigcup_{i=0}^{N_{\mathrm{sn}}-1} \bigcup_{s=0}^{N_{\mathrm{se}}(i)} v_a(i, s) \ . \tag{3}$$

In Figure 2(c) the white and black nodes are the set V_a.

Interior nodes are then added by spherical linear interpolation between v_c and the vertices in V_a. The required number of nodes for each edge is given by

$$N_{\mathrm{si}}(i, s) = \lceil (r_{bs} + r_e) \arccos\left(v_a(i, s) \cdot v_c\right) / l_{\max} \rceil \ .$$

The vertex for each interpolation is then

$$n_{sp}(i, s, t) = p\big(v_a(i, s), v_c, N_{\mathrm{si}}(i, s)\big) \ , \tag{4}$$

and the total set of vertices for the spherical patch is

$$V_s = v_c \cup \bigcup_{i=0}^{N_{\text{sn}}-1} \bigcup_{s=0}^{N_{\text{se}}(i)} \bigcup_{t=0}^{N_{\text{si}}(i,s)} n_{sp}(i,s,t) \ . \tag{5}$$

The interior nodes in this set are shown in Figure 2(c) as green nodes. The interpolation functions above have no singularities since the angular spacing between consecutive vertices are always less than π for any obstacle that extends three dimensions.

2.1.2 Cylinder patch vertices

From the sphere patch a set of vertices was found at the arc edges of the cylinder patch. Both arc edge has the same number of vertices, since the edges have the same length and are both interpolated using $l_{\max}$. Assume now that the vertices v_i at one end is paired one-to-one with the vertices w_i at the other end, in an ordered manner starting at the same facet edge. Since the height of the cylinder is constant, the number of vertices along this height must be

$$N_{\text{c}} = \lceil ||w_0 - v_0||/l_{max} \rceil \ .$$

Each interpolation point for pair i is given by

$$n_{\text{cp}}(i,j) = v_i + (w_i - v_i)j/N_{\text{c}}$$

for $j = 1, \ldots, N_{\text{c}} - 1$ and for $i = 0, \ldots, N_{\text{se}}(u)$, where u is the index of the corresponding sphere patch polygon edge. Now the set of all cylinder patch vertices is given as

$$V_c = \bigcup_{i=0}^{N_{\text{se}}(u)} \bigcup_{j=1}^{N_{\text{c}}-1} n_{\text{cp}}(i,j) \tag{6}$$

2.1.3 Full obstacle visibility graph

The VG for the obstacle is now given as the union of the sets V_s in (9) and V_c if (12). The links of each patch is found by connecting all its vertices. Links between vertices on the same facet edge can be removed unless both are endpoint nodes, since such links will not be part of the shortest path.

2.2 Via Points Visibility Graph

When generating a VG between a via point (initial or desired vehicle position) and the obstacle nodes, only links on supporting lines are part of the shortest path and hence included.

In order for a link L_n from a node n_e to a node n_s on a polyhedron surface S to be on a supporting line, at least one of the N_f facets of n_s must be 'visible' and at least one facet must be non-'visible'. A facet k of n_s is visible from node n_e if

$$f_v(n_e, n_s, k) = (n_e - n_s) \cdot g(k)$$

is ≥ 0. The function $g(k)$ gives the outward-pointing normal vector for a facet k of n_s. This means that L_n is a on a supporting line iff there exists $j = 0, \ldots, N_f - 1$ such that $f_v(\cdot, \cdot, j) \geq 0$ and there exists $k = 0, \ldots, N_f - 1$ such that $f_v(\cdot, \cdot, k) < 0$.

We now define the full graph as

$$\mathrm{VG}_{\mathrm{vp}} = \mathrm{tg}(\mathrm{vp}_0, \mathrm{VG}_o) \cup \mathrm{tg}(\mathrm{vp}_1, \mathrm{VG}_o) \cup \mathrm{int}(\mathrm{vp}_0, \mathrm{vp}_1) \ ,$$

where $\mathrm{tg}(p, G)$ is a function that gives the supporting VG from the supporting link set between p and G, and $\mathrm{int}(p_0, p_1)$ is a function that returns the VG between p_0 and p_1, if its link does not intersect the CS obstacle. Such a function can be based on segment/triangle intersection, see e.g. [10].

3 RESULTS

An example is created by generating a VG from the vertices of an obstacle shaped as a building. The setup and results can be seen in Figure 3. A small-scale helicopter with radius r_{bs} of 1.70 m is used. The interpolation parameter $l_{\max}$ is set to 0.75 m and r_e to 0.076 m. The VPs are located at $(-1, 0, -11)$ [m] and $(0, 2, 10)$ [m], and a path is calculated by searching the VG. This approximated shortest path p_a is then compared to the optimal path. The optimal path p_g is composed of geodesics and tangents on the CS obstacle and is calculated using a minimization algorithm.

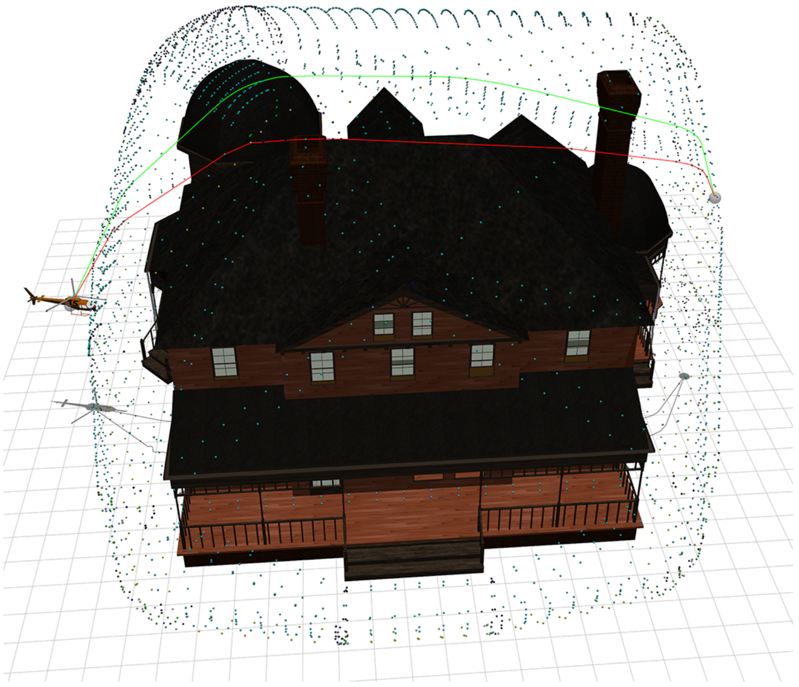
The building consists of 33 thousand vertices and 59 thousand triangles. The resulting WS obstacle has 128 vertices and 182 facets. In CS this obstacle consists of 182 facets, 128 sphere patches, and 308 cylinders patches. A VG was generated in 551 miliseconds[1] that has 4 thousand nodes and 53 thousand links. A path was found in 457 miliseconds by searching the VG with Dijkstra's graph search algorithm.

Figure 3 shows how a (local) minimum on one side of the chimney has been found, whereas the optimal path is located on the other side. Although different paths have been chosen, both have nearly same length. The optimal path is 27.4 m while the approximated path is 27.9 m, giving an increase in distance of less than 2%. To improve the solution, the interpolation parameter $l_{\max}$ can be reduced. To generate a smaller VG that is faster to search, $l_{\max}$ can be increased. However, theres is a limit on how small the VG will become for a given obstacle model. An example is the dome in the left part of the building in Figure 3(b). Since all points on the sphere are part of its convex hull, a dome will add many smaller patches to the CS, which results in a higher density of nodes. To overcome this, a polygon reduction algorithm can be applied to the model before generating the workspace.

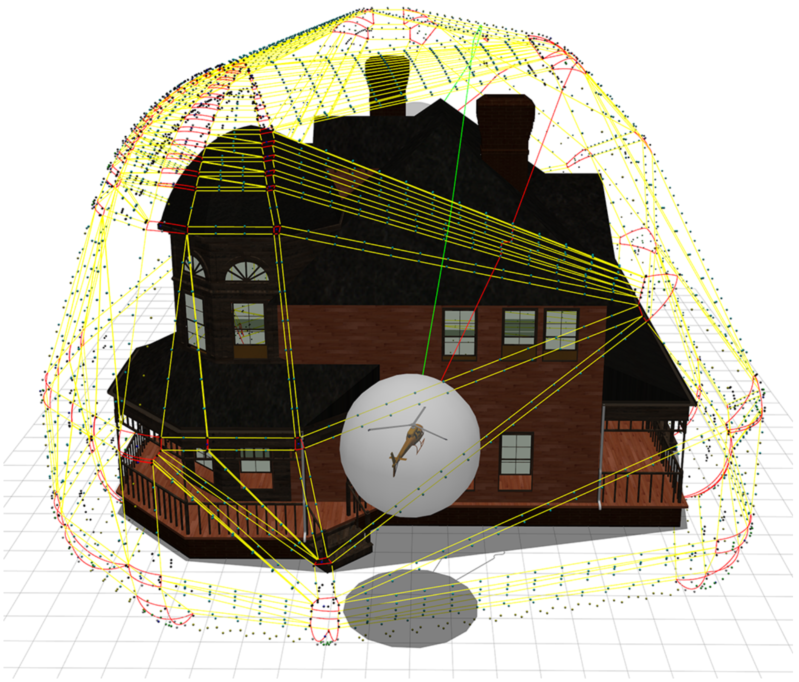
4 CONCLUSIONS

Finding an optimal solution to the path planning problem in 3D is NP-complete. The shortest path around a polyhedral configuration space obstacle does not in general traverse only vertices of the polyhedron, as in the 2D case, but also points on its edges. For a given resolution the presented method gives

[1]All tests were done on a single core of a 2.2GHz Intel Core 2 Duo laptop



(a) Front view of building



(b) Side view of building. The CS is shown by the outline of the translated facets and patches are also drawn. Facet are drawn in yellow, sphere patches in red, and cylinder patches in alternating red and yellow. The helicopter is drawn with its bounding sphere.

Figure 3: All nodes in the VG as drawn as small spheres. The larger spheres show the VPs, between which the red approximated path and the green optimal path is drawn.

a path for which the length is close to optimal, and converges towards optimal by increasing resolution. This means that the proposed method might find a path that takes different route than the optimal path, but the approximated path will be almost as short as the optimal.

We proposed a method that generates a VG that is improved for finding a shortest path in an environment with a single obstacle. The applied method is based on first constructing the CS obstacle from the WS obstacle described by a point-cloud, then generating an obstacle VG for the CS obstacle that approximates its surface. This VG is combined with VGs that links the obstacle VG to the initial and desired configuration. If a path exist that connects the initial to the desired configuration, it can be found using a graph search algorithm.

Extending the method to work with multiple via points and multiple obstacles is possible, though the latter would require intersection testing of link and node candidates in the VG.

REFERENCES

[1] M. A. Wesley T. Lozano-Perez. *An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles.* Communications of the ACM, 22, 1979.

[2] J. H. Reif J. Canny. *New lower bound techniques for robot motion planning problems.* In Proc. 28th Annu. IEEE Sympos. Found. Comput. Sci., 1987.

[3] T. Lozano-Perez. *Spatial Planning: A Configuration Space Approach.* IEEE Transactions on Computers, 32, 1983.

[4] M. Sharir E. Oks. *Minkowski Sums of Monotone and General Simple Polygons.* Discrete and Computational Geometry, 2006.

[5] M. Kim C. Bajaj. *Generation of configuration space obstacles 1: the case of a moving sphere.* IEEE Journal of Robotics and Automation, 1988.

[6] E. W. Dijkstra. *A note on two problems in connection with graphs.* Numerische Mathematik 1: 269-271, 1959.

[7] B. Raphael P. E. Hart, N. J. Nilsson. *A Formal Basis for the Heuristic Determination of Minimum Cost Paths.* IEEE Transactions on Systems Science and Cybernetics SSC4: 100-107, 1968.

[8] J. O'Rourke. *Computational Geometry in C.* Cambridge University Press, 1998.

[9] J. Fillmore S. R. Buss. *Spherical Averages and Applications to Spherical Splines and Interpolation.* ACM Transactions on Graphics 20, 2001.

[10] B. Trumbore T. Moller. *Fast, Minimum Storage Ray-Triangle Intersection.* J. Graphics Tools 2(1), 1997.