Configuration Space and Visibility Graph Generation from Geometric Workspaces for UAVs

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

Published in:
AIAA Guidance, Navigation, and Control Conference 2011

DOI (link to publication from Publisher):
10.2514/6.2011-6416

Publication date:
2011

Document Version
Early version, also known as pre-print

Configuration Space and Visibility Graph Generation for UAVs from Geometric Workspaces

Flemming Schøler* and Anders la Cour-Harbo† and Morten Bisgaard‡
Aalborg University, 9220 Aalborg East, Denmark

We address the challenge of generating a visibility graph for 3D path planning for a small-scale helicopter through an environment with geometric obstacles. The visibility graph is based on approximating a number of continuous differentiable configuration space obstacles that are generated from the convex hull of their work space obstacles. An approximation to the optimal path can subsequently be found using an existing graph search algorithm. The indented use of our approach is in well-known environments. Our approach is used in a test scenario for generating a path for a small-scale helicopter inspecting a wind turbine.

I. Introduction

UAS have become a preferred, indispensable, and increasingly used platform for many applications where manned operation is considered unnecessary, repetitive, or too dangerous.[1,2] Path planning and trajectory generation are fundamental area for UAS development in general[3] and is used to find a path through an environment under a set of constraints. Specifically, for tasks such as surveillance, inspection, search and rescue, aerial mapping, cinematography, 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. Methods for describing these obstacles as geometric figures from real-world measurements have also been improved recently[27,28] with 3D environment mapping data getting increasingly detailed and readily available including entire city maps. These recent advances now allows the use of 3D maps that are fairly accurate for larger static obstacles that can be used generally for high-level planning in largely static environments, such as a city, or for low-level planning in fully static environments, such as inspecting a stopped wind turbine.

A. Background

Here we consider methods for path planning that can be used to operating a small-scale helicopter near the surface of obstacles. We distinguish the term "motion planning" from "path planning" in that the computed path is parameterized by time (i.e. a trajectory). In this paper we are manly concerned with geometric problem of path planning. We see the general motion planning problem for UAVs as having multiple levels, where the top level is focused on planning a path that avoids larger static obstacles, e.g. buildings, mountains, trees. Because of this level being static, we also prioritize the ability to do multi-query searches. We look at feasibility of the path from a qualitative perspective by preferring path with higher order of differentiable. This means that we do not consider the structural and dynamical capabilities of the vehicle in greater details at this level. This and dynamic obstacles should be handled at a lower trajectory planning level, that generates full state and control trajectories, and typically also does single query searches because of the dynamic obstacles. Although we only look at top level path planning, we still employ a stepwise strategy, that more efficiently can handle isolated changes in the environment as well as re-planning where only changes to via points change.

This work is focused on path planning using visibility graphs that is based on the configuration space. The generalized visibility graph $VG(N, L)$ contains a node set N and a link set L of links between node pairs. The nodes are sampled from the edges of the configuration space, and will hence only approximate

*Ph.D. Stud., Institute of Electronic Systems, Fredrik Bajers Vej 7C.
†Assoc Prof., Institute of Electronic Systems, Fredrik Bajers Vej 7C.
‡Assist Prof., Institute of Electronic Systems, Fredrik Bajers Vej 7C.

an optimal solution. The link set is directed and produced according to a custom optimality criteria. This means that other costs than Euclidean distance can be used and links can have two different costs according to the direction of travel. To obtain a path, the visibility graph should be searched using a graph search algorithm, for instance Dijkstra's algorithm[23] or A*.[24] However this part is not the focus of this research, and we will not go into details. Contrary to other popular methods such as the Rapidly-exploring random tree method, searching the tree will also reveal if no solution exists.

Path planning using visibility graphs has been subject to much research. In comparison, the literature on how to practically, automatically and efficiently generate configuration spaces from a known general 3D work space polyhedron seems limited to building primitive solids or geometric representations[7] and often an existing configuration space is assumed. The main contribution of this paper lay in an algorithm that build geometric configuration spaces and a visibility graph from a group of known workspace obstacles. Our algorithm works on a collection of meshes or point-cloud representations of the obstacles, and exploits knowledge of the obstacles to efficiently determining links for the graph.

In order to meet that goal some generalizations must be made. The configuration space for a helicopter has usually six degrees-of-freedom, with three dimensions to describe position and three to describe attitude. However, by considering a bounding sphere for the helicopter that is centered in center-of-mass, the attitude will be irrelevant when determining a collision free path and the configuration space can be reduced to three degrees-of-freedom. Besides reducing the complexity of the problem (still a NP-complete), the configuration space obstacles will also get a continuous differentiable surface and any geodesic path will be continuous differentiable which will be easier to track. Generally the higher order of smoothness our path has the easier it should be to track. Another advantage is that one issue criticized about the visibility graph method can be easily avoided. This issue is that it is impractical to operate near obstacles since errors will cause collisions. This issue is avoided by introducing a safety margin that increases the vehicle bounding sphere.

B. Previous Work

Visibility graphs for path planning has been used extensively, however finding an optimal solution to the general path planning problem in 3D is NP-complete.[11,12] 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 its edges. The concept of adding additional vertices along these edges in configuration space so that no edge is longer than a specified maximum length is introduced by.[11] This method generally results in a good approximation to the optimum path.

In[15] an algorithm is presented that generates a configuration space obstacle in 2D from a convex polygonal obstacle and a convex polygonal vehicle. This is done by sliding the vehicle polygon around the obstacle in such a way that they are always in contact at one point. This approach would be adapted to our case by approximating the vehicle by for instance a geodesic dome. While this algorithm can be extended to higher dimensions it results in semi-algebraic models for configuration space obstacles and is already overly complicated for our purpose in its 2D form.

Another common approach used for polyhedron is based on convex hulls. For a convex polyhedron vehicle and obstacle, it calculates the Minkowski sum polyhedron by performing vector addition of all points of the vehicle and obstacle and computing the convex hull of the resulting point set. Convex hulls are computed by several algorithms such as: the Quickhull algorithm,[16] the gift-wrapping algorithm,[19] the incremental algorithm,[17,18] the divide and conquer algorithm,[20] and Graham's algorithm.[21]

We suggest a similar approach, but substitute the convex polygonal of the vehicle with a sphere approximation. This leads to a 3 degrees of freedom configuration space, and a configuration space that is going towards continuous differentiable as the resolution of the approximation increases. A continuous differentiable work space allows us to find paths that are easier to track for a physical system than the discontinues ones of the former approach, because it does not require instantaneous changes in velocity (but in acceleration).

C. What we do

In this paper the focus is on inspection tasks where helicopter must stay close to obstacle surface with multi-query searches must be done due to i.e. multiple via points or re-planning. The focus is also on cases where limited changes occur in the workspace.

In this paper we consider the configuration space of an, at least conceptually, spherical vehicle moving amongst polyhedral obstacles, i.e. obstacles described by patches of planar surfaces. We generate the configuration space by translating or growing each surface patch by the radius of the sphere. This approach will work for surfaces of degree $\leq 2^{25}$ only, i.e. cylindrical and spherical surfaces.

Our approach will generate a convex configuration space surface for each obstacle. This surface goes towards continuous differentiable as resolution improves. We take an sequential approach by 1) construction a visibility graph for each obstacle, 2) then prune nodes and links from intersecting obstacles, 3) combine the visibility graphs, 4) finally include nodes and links from via points. The advantage of this sequential approach is that while all steps must be completed when creating the initial visibility graph, only a partial update of the visibility graph is required when obstacles are added or removed, even less when obstacles are just moved or rotated, and even further less when just via points are changed.

II. METHOD

The general problem of finding a path for a helicopter is a 6 degrees-of-freedom problem. Our approach generates a configuration space based on the Minkowski sum (see[14] for details on Minkowski sum) of the vehicle bounding sphere and the convex hull of work space obstacles. This approach makes the configuration space invariant to vehicle orientation, and thus reduces the problem to a 3 degrees-of-freedom problem. These obstacles are represented in configuration space as a set of primitives (planes, cylinders, and spheres) generated by the algorithm from the any work space obstacles.

A. Configuration Space

The space of all possible positions, denoted the configuration space, is constrained by the obstacles. For each obstacle in work space, a configuration obstacle is created. The configuration space obstacle is the Minkowski sum of the vehicle bounding sphere and the convex hull of a work space obstacle polyhedron. From a geometric point of view, this Minkowski sum is obtained when vertices in the convex hull of the workspace are replaced with sphere patches, edges with cylinders, and faces are translated along their normal. An example of this is shown in Figure 1.

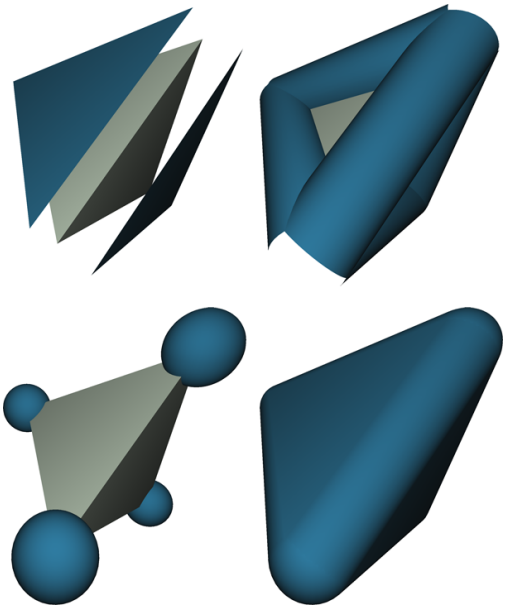

Figure 1. Configuration space is formed by the Minkowski sum of a polyhedron workspace and a sphere: Facets are translated, cylinders are drawn at edges, and spheres are drawn at vertices.

We decompose the configuration space obstacles by patches of primitive surfaces. These primitives are sphere, cylinder, and planar facets. Any two neighboring patches intersect at one common edge. This edge is either straight or an arc edges. It follows from the Minkowski sum operation that spherical patches

only has cylindrical patches as neighbors. Cylindrical patches have two opposing sphere patches and two opposing facet patches as neighbors. Facet patches only have neighboring cylindrical patches. Because of this composition, all nodes for the visibility graph can be determined from the sphere and cylinder patches. These nodes origin from the interior of the sphere and cylinder patches, and from their common edge.

Once the node set is formed for a patch, the link set can be easily determined knowing that the obstacle is convex. This means that links between nodes belonging to an obstacle are restricted to nodes of the same patch.

The Sections B and C explain how the node set is formed based on the sphere and cylinder patches and how the link set is found on each of the three patch types. In Section D we extend our visibility graph to include multiple obstacles.

B. Sphere Patch

A sphere patch was added to the configuration space obstacle for each vertex in workspace obstacle. A sphere patch C is a simple closed geometric figure on the surface of a unit hemisphere that is formed by the arc segments on the great circles (the geodesics) between N cyclic ordered vertices v_i, $i = 0$ to $N-1$. The last vertex v_N is assumed to be the same as the first, i.e.: the patch is closed. The sphere patch is convex so the geodesic between any two vertices in C lies entirely in C. Each vertex in the patch is obtained by translating a workspace vertex along the neighboring facet normal in an ordered matter. So the patch polygon has the same number of nodes as number of facets that use the workspace vertex. The subject of ordering the vertices is fairly standard and will not be covered here. Each vertex is shared by two cylinder patches, one sphere patch and one planar patch. Each segment of the patch neighbor a different cylinder patch. The objective is now to determine where to add nodes and links for each patch, such that as few nodes as possible are used in the visibility graph, while at the same time limiting the volume of the configuration space obstacle. Since any two different points on the patch will not have line-of-sight, it is necessary to place them above the surface.

We reduce the spherical polygon that outline C to a set of connected spherical triangles to easier determine where to add nodes. We let the spherical polygon be composed by N spherical triangles, where each triangle shares a shares an interior central vertex denoted q. Because of convexity this interior vertex is given by the weighted average in Eq. 1 of the vertex set v and some non-negative weights w_i that sum to 1.

$$q = \sum_{i=0}^{N-1} w_i v_i \tag{1}$$

A central vertex candidate can be found simply by equating all weights to $1/N$, but a better choice is usually to find the central vertex that minimize spherical distance to the vertex set. This spherical weighted average is equal to the point q on the unit sphere S which minimize $f(q)$;

$$f(q) = \frac{1}{2} \sum_{i=0}^{N-1} w_i \mathrm{dist}_S(q, v_i) \tag{2}$$

where $\mathrm{dist}_S(p, q)$ is the geodesic distance from p to q on S. This problem is well-defined for vertices on a hemisphere and a fast iterative algorithm with quadratic convergence rate exists, see.[29]

The visibility graph, that is the node set and link set, for the sphere patches must be located above/outside the patch, so that neighboring nodes will have line-of-sight (i.e. not intersecting the patch). In essence the visibility graph takes shape of a triangle mesh approximating the sphere patch. How far above the nodes must be located are resolution dependent - the greater resolution, i.e. nodes, the less displacement is required. Here we base the calculation of this displacement r_n on the worst case:

$$r_n = \frac{r_{bs}}{\sqrt{0.5\cos(a_{max})^2 + 0.5}} \tag{3}$$

where:
a_{max} is the worst case angle between two neighboring nodes. It's absolute value is less than pi/2.
r_{bs} is the vehicle bounding sphere radius
r_n is the required radius

1. Node Set

In order to construct the node set for the obstacle, we first interpolate the edge endpoints of the spherical polygon to find a set of vertices v_a. We then interpolate the edges formed by these vertices and the central vertex to get a vertex set, that can be scaled by r_n to obtain the nodes.

From the set of vertices v that consecutively form the edges of one spherical polygon. The number of nodes $N_s(i)$ along an edge index i required for a certain resolution, determined by l_{max}, is given by:

$$N_s(i) = \left\lceil \frac{r_n \cos^{-1}(v_i \cdot v_{i+1})}{l_{max}} \right\rceil \tag{4}$$

where l_{max} is the maximum allowed arc distance between neighboring nodes.

The vertex set $v_a(i,s)$ on the arc between v_i and v_{i+1}, and where $s = 0$ to $N_s(i) - 1$ is:

$$v_a(i,s) = \frac{\sin\left[\left(1 - \frac{s}{N_s(i)}\right)\cos^{-1}(v_i \cdot v_{i+1})\right] v_i + \sin\left[\frac{s}{N_s(i)}\cos^{-1}(v_i \cdot v_{i+1})\right] v_{i+1}}{\sqrt{1 - (v_i \cdot v_{i+1})^2}} \tag{5}$$

Nodes are then added by spherical linear interpolation between the central vertex v_c and the vertices in v_a for each edge. The required number of nodes is given by $N_t(i,s)$:

$$N_t(i,s) = \left\lceil \frac{r_n \cos^{-1}(v_a(i,s) \cdot v_c)}{l_{max}} \right\rceil \tag{6}$$

where l_{max} is the maximum allowed distance between neighboring nodes.

The node set $n_{sp}(i,s,t)$ is then given for integer values of $t = 0$ to $N_t(i) - 1$ as:

$$n_{sp}(i,s,t) = r_n \frac{\sin\left[\left(1 - \frac{t}{N_t(i)}\right)\cos^{-1}(v_i \cdot v_c)\right] v_a(i,s) + \sin\left[\frac{t}{N_t(i)}\cos^{-1}(v_i \cdot v_c)\right] v_c}{\sqrt{1 - (v_i \cdot v_c)^2}} \tag{7}$$

It should also be noted that the interpolation functions above have no singularities since consecutive vertices are always spaced less than pi for any obstacle that extend in three dimensions. No node is added at the central vertex v_c by the equations above. This node is given by:

$$n_{sp.p} = r_n v_c \tag{8}$$

2. Link Set

The link set is found from the set of edges defining the convex hull of the node set above the sphere patch. The convex hull of a set of points S is the intersection of all convex sets containing S. For N nodes p_1, ..., p_N, the convex hull C is then given by the expression:

$$C = \left\{ \sum_{i=1}^{N} \lambda_i n_i : \lambda_i \geq 0 \ \forall \ i, \sum_{i=1}^{N} \lambda_i = 1 \right\} \tag{9}$$

As stated earlier, numerous methods exits for finding the convex hull, see e.g. O'Rourke[22] for a robust $O(n^2)$ three-dimensional implementation and Qhull[16] also works efficiently. Since a sphere patch is connected to its neighbors, it should be an open polygon, whereas the hull obtained by the mentioned algorithms is closed. One way of handling this is to add a dummy node is on the opposite hemisphere, e.g $-v_c$ before calculating the hull. This node and connected edges can then be removed after the hull is found, which result in an open polygon that can be patched with its link set from the neighboring cylinder patches.

C. Cylinder Patch

From the sphere patch a set of nodes was found at the arc edges of the cylinder patch. We now add a link for each opposing node pair and possible further nodes to improve resolution.

We then connect opposing and neighboring nodes into rectangular facets. Additional nodes are inserted along opposing nodes, so that no nodes are spaced more than a maximum distance l_{max} in each facet of the path:

$$N_j(i) = \left\lceil \frac{||v_{i+1} - v_i||}{l_{max}} \right\rceil \tag{10}$$

$$n_{cp}(i,j) = v_i + (v_{i+1} - v_i)\frac{j}{N_j(i)} \tag{11}$$

where
$N_j(i)$ is number of segments along edge i, where $i = 0$ to $N - 1$.
$n_{cp}(i,j)$ is added nodes along edge for planar patch, where $j = 1$ to $N_j(i) - 1$ since we exclude endpoints.
The link set is generated by connecting nodes after the rule describe in Figure 2.

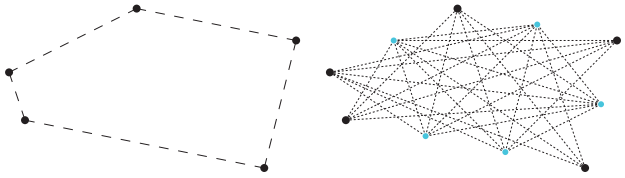

Figure 2. Each facet adds links to the visibility graph. Nodes are added at the vertices of the facet and along its edges, so that two neighboring nodes on an edge are not spaced more than a given threshold. Links are added between all pairs of nodes that are not located on the same edge unless the pair is the endpoints of that edge. This pruning is done since such links will not be part of an optimal path. The left picture shows the visibility graph constructed from the edges and vertices of the facet. The right picture shows where additional nodes and links are added. Edges and vertices along the facets are generally shared by neighboring facets but are only added to the graph once.

The number of segments an edge is split into is given by:

$$N_v(i) = \left\lceil \frac{|v_i - v_{i+1}|}{l_{max}} \right\rceil \tag{12}$$

At this point all nodes for the obstacle visibility graph has been found, and only the link from of the planar patches remain. These links are again found by connecting nodes after the rule describe in Figure 2.

D. Composing a Visibility Graph

Once visibility graphs are generated for each obstacle, the via points are added, and node pairs from different obstacles are added to the graph is they have line-of-sight. To test line-of-sight an intersection test is performed that tests against all obstacles for intersection, until either an intersection is found or all obstacles have been evaluated.

Methods for collision testing are numerous. One method is to test the candidate link edge for intersection against all facets of an obstacle, which can be done by first determining the point of the segment/plane intersection and then determining if this point is inside the face polygon. Testing whether a point is inside a polygon is a common operation in computer graphics and several methods are compared by.[30] For cases with less than 10 edges per face, algorithms which partition the polygon into triangles are amongst the fastest. Not all triangles necessarily have to be tested. Creating a spatial search tree can significantly reduce the number of tests required and hence the computational load. This works by using a faster test to initially exclude groups of faces that do not intersect the segment.

III. Results

The implemented method for construction visibility graphs will be demonstrated by finding a path around a wind turbine that visits a number of via points near the surface as shown in Figure 3. The wind turbine is a Vestas V52 model. The helicopter model is a Bergen Industrial Twin helicopter that has a bounding sphere radius of 1.7 m. This test case will take the helicopter through six via points (vp). The vps are listed in Table 1.

Table 1. Position of the six via points.

Via point	1	2	3	4	5	6
X (m)	-2.0	4.0	-3.5	8.5	-6.0	-3.5
Y (m)	-4.0	1.0	2.0	-2.0	-32.5	-4.0
Z (m)	1.0	35.0	68.0	69.5	53.0	67.0

The path must start off at the base of the turbine (vp 1) and moves towards the turbine house (vp 3). Halfway up the tower a via point at the back of the tower is visited (vp 2). From the turbine house the path moves underneath the nose between two rotor blades to a position on the base of one the rotors (vp 4). This rotor is inspected by moving towards a point on its tip (vp 5) and then back to the base, this time one the back of the rotor (vp 6).

Instead of showing the full visibility graph, which is more difficult to illustrate in 3D, we usually only show the combination of nodes and links that result in the optimal path. This path is determined by a graph search algorithm.

A. Preparing the work space

The wind turbine model seen in Figure 3 is composed of a tower, a gearbox house, and three rotors attached to a nose. Since the obstacle visibility graphs will be based on the convex hulls of the work space obstacles, a more accurate visibility graph can be obtained by operating on each part of the model. An obstacle visibility graph is hence build for each part before they are combined to the final visibility graph.

In this test scenario we have tried to stay true to the original model, by not make some of the generalizations that would result in a simpler work space and subsequent simpler problem. A cylindrical part could be represented by a single cylinder primitive, by adding the radius to the primitive in configuration space. However, in this case the cylindrical tower has slightly different top and bottom radius, so it has to be represented by multiple primitives. If the gearbox house did not have rounded edges there would be fewer nodes in the visibility graph.

While the path planning algorithm can handle a visibility graph with a large number of nodes and links, an improvement in speed can be obtained by reducing the number of polygons in each part. A polygon reduction algorithm has been used on most parts to slightly reduce the number of nodes in the model. A side effect of this process is the removal of some of the symmetry of the original model. Because the rotors are different only in position and orientation, each rotor has the same shape, and only one obstacle visibility graph needs to be constructed for the three rotors. This approach does not however change the node or link count in the final graph.

B. Searching Visibility Graph

Once given a visibility graph we aim at finding the minimal cost path. This path is found by search through the graph with a cost associated to each edge in the graph. In this scenario, the Euclidean distance is used, making this approach independent of vehicle dynamics, but other criteria for optimization of the path could be used in the visibility graph. Since the visibility graph is constructed in 3D it is based on approximation through a finite number of points located on surfaces of the configuration space obstacles. This in unlike the 2D case where (at least for obstacles with non-differentiable edges) a finite set of nodes (all 'corners' of all obstacles) is sufficient to guarantee an optimal path. Thus, the line-of-sight trajectory is only locally shortest among the vertices in the visibility graph, and not globally shortest as it would be in the 2D case. However, given a sufficiently dense sampling of the 3D obstacles a close to optimal path can be found.

In the sections below the entire path is split into three parts: Moving along the tower in vp 1-3, moving under the nose in vp 3-4, and moving around a rotor in vp 4-6.

Vp 1-3 The path starts off from the base of the tower and moves upwards towards a point half way up the tower before continuing to the gearbox house. The half way point is not actually located on the configuration space surface. From the two plots in Figure 4 a good consistency between the geodesic and approximated path can been seen. The largest discrepancy can be seen in the top view near the vp 3. Where the geodesic path can leave the configuration space surface as soon as it "sees" vp 3, the approximated path must wait

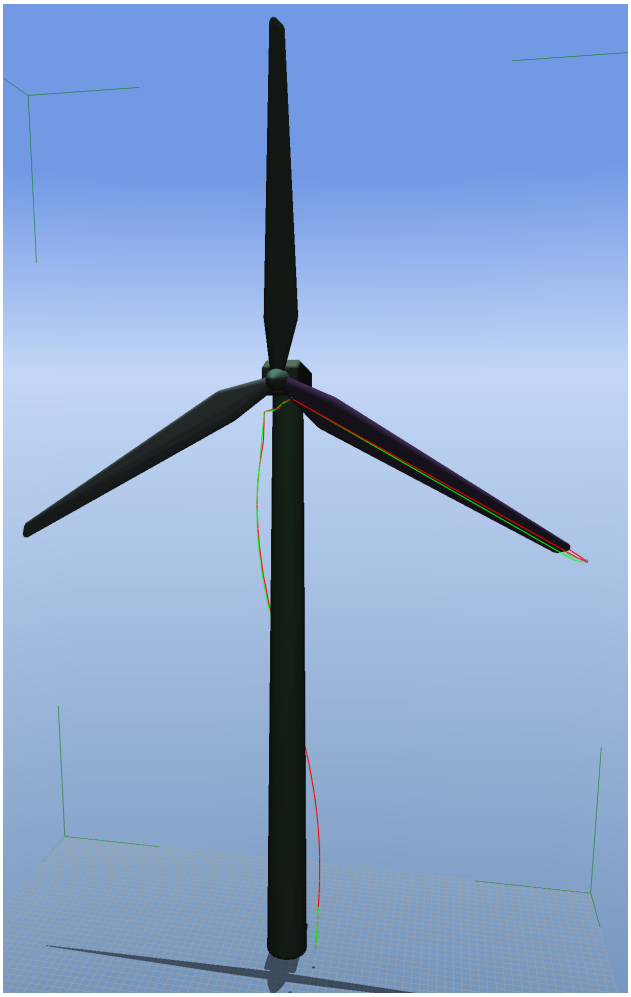

Figure 3. Front view of wind turbine.

until until a node is found that has a visible link to the vp. As a result, the path tends to continue too far along the surface before moving towards the vp. From the top view it can also be seen that the radius of the tower at the base is larger than at the top, since both paths intersects the y-axis at -4.2 m near the base while intersecting 3.6 m near the top. As a result the geodesic path on the tower configuration space surface is not a straight line on the side view. The largest difference between the paths are between vp 2 and

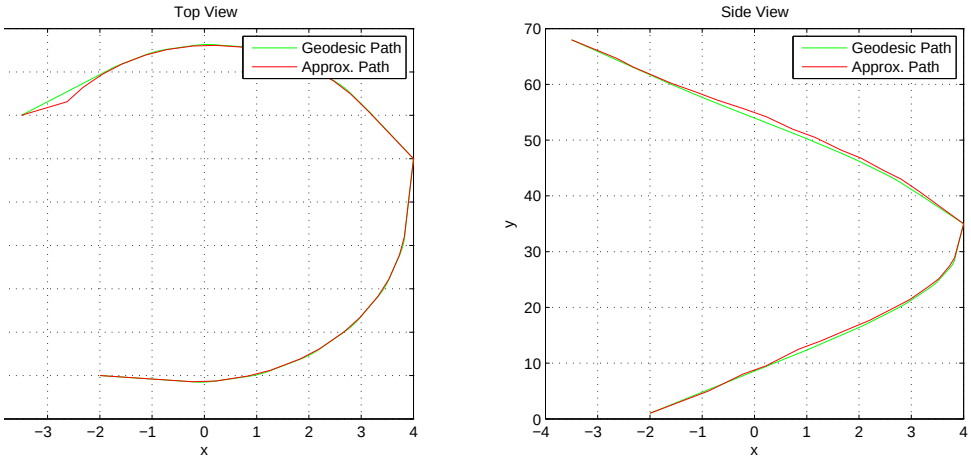

Figure 4. Side and top plot of path between vp 1 and vp 3.

vp 3 the side view. Figure 5 shows a close up of some of this path. It seems that although nodes could have been picked that seem nearer to the geodesic path, this does not result in a globally shorter path. When comparing the length of the two paths, the approximated path is only 0.05 % longer. Both paths can be seen in Figure 6 that also show the location of the tower and the patches on the tower. The patches at the top and bottom are located inside other obstacles or outside bounds, and do not add to the visibility graph.

Vp 3-4 The path moves between the two downwards pointing rotors to a position in front of one of the rotors. The two plots in Figure 7 show a larger error between the paths and the approximated path is 2.1 % longer. Because the geodesic method can only operate on one obstacle at a time, we use the approximated path to add intermediate via points on the approximated path in between via point 3 and via point 4, such that each part of the geodesic path is only located on a single obstacle. Although this means that the geodesic path is only optimal between the intermediate via points, we find this sufficient for comparison. The location of the two added via points can be seen on Figure 8, as well as the relevant visibility graph link set for each obstacle. The links set interconnecting different obstacles are too many to show here. Links and nodes in the obstacle visibility graph that violates another configuration space are removed when combining the visibility graphs. Ideally new nodes and links should be added where these obstacles intersect, since they form ridges where local minima are likely. This should result in a better approximation of the shortest path, but is left as an issue to be addressed in further work.

Vp 4-6 The path between vp 4 and 6 moves from the front of the a rotor to the tip, and continues to a point behind the rotor. The approximated and geodesic path can be seen both in Figure 9 and Figure 10. Although the approximated path is somewhat different from the geodesic one they have similar length. The approximated path is just 0.06 % longer than the geodesic path. Since there might be several local minima for the path planning problem, the approximated path might be entirely different and still be nearly as short at the geodesic path.

IV. Discussion

Finding an optimal solution to the path planning problem in 3D is a NP-complete problem. Since 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. If one suffice with an approximated

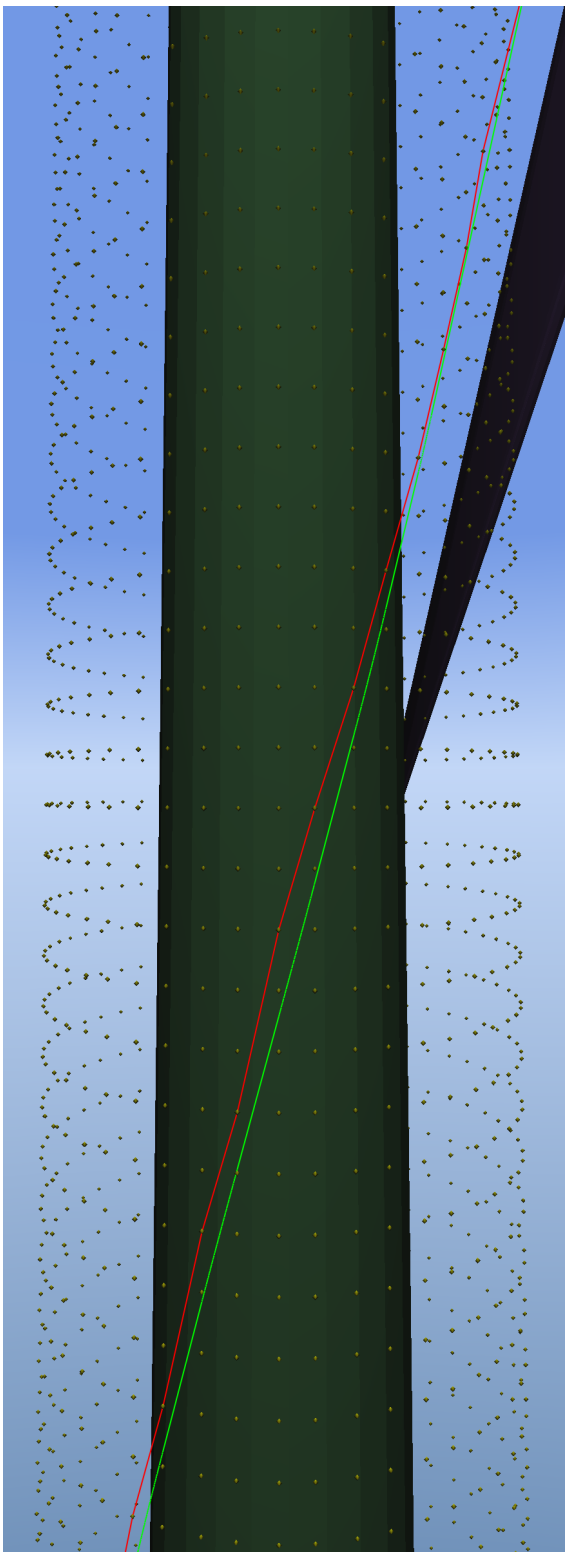

Figure 5. Section of path between vp 2 and vp 3. The approximated path is drawn in red and the geodesic path is drawn in green. The visibility graph nodes for the tower obstacle are drawn as yellow diamonds.

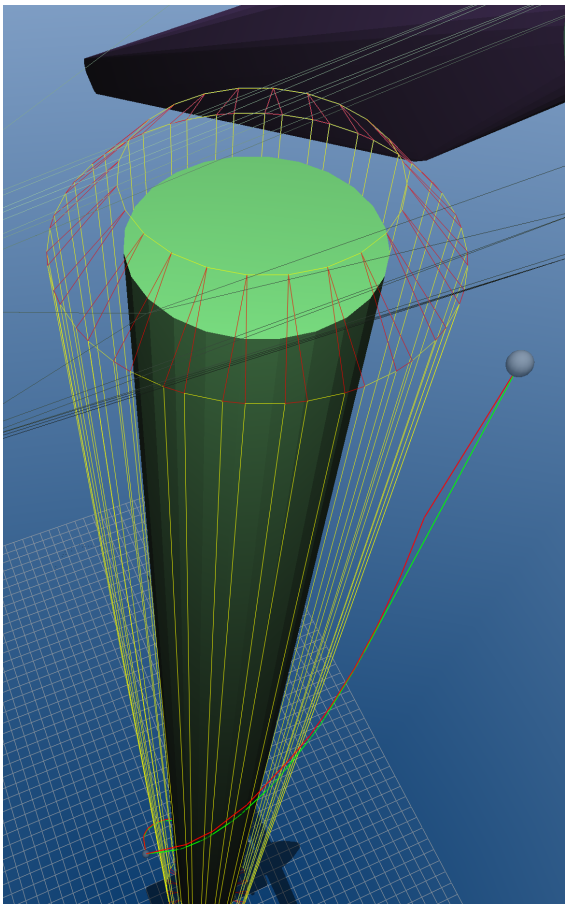

Figure 6. Tower and the domains for its different patches. The gearbox house is viewed as a wire-frame to give better view of the patches. Rectangles with alternating colors show cylinder patches, red triangles show sphere patches, and yellow polygons show the facet patches. The approximated path is drawn in red. The geodesic path is drawn in green. Via points are indicated with a gray sphere. Only vp 2 and 3 are visible on this picture.

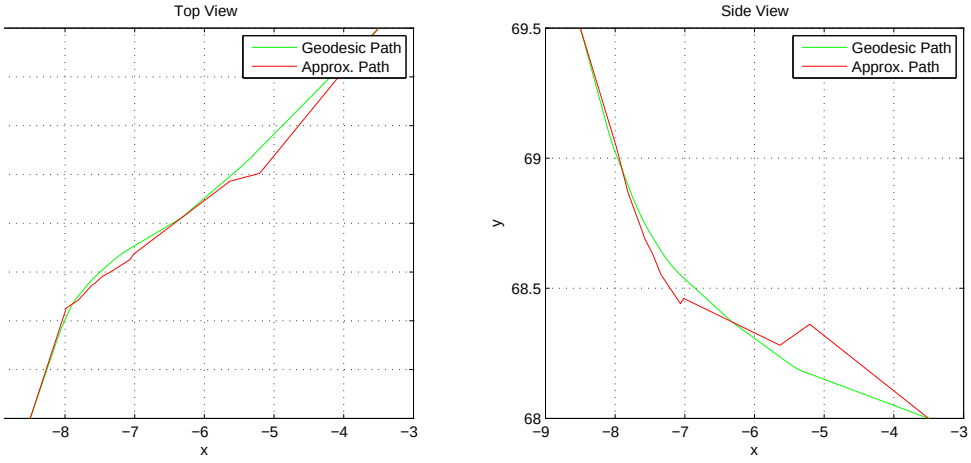

Figure 7. Side and top plot of path between vp 3 and vp 4.

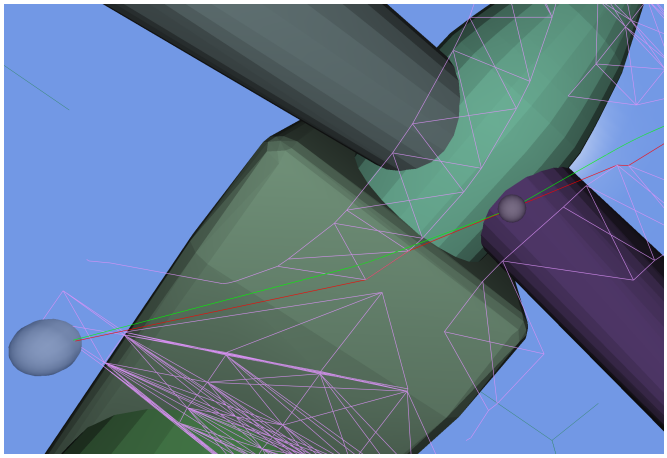

Figure 8. Path between vp 3 and vp 4. The approximated path is drawn in red and the geodesic path is drawn in green. A section of the link set is drawn for each obstacle.

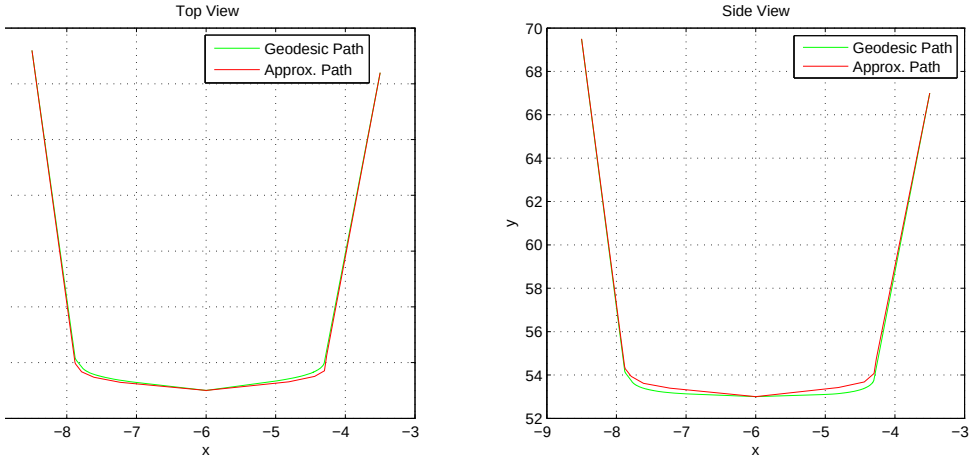

Figure 9. Side and top plot of path between vp 4 and vp 6.

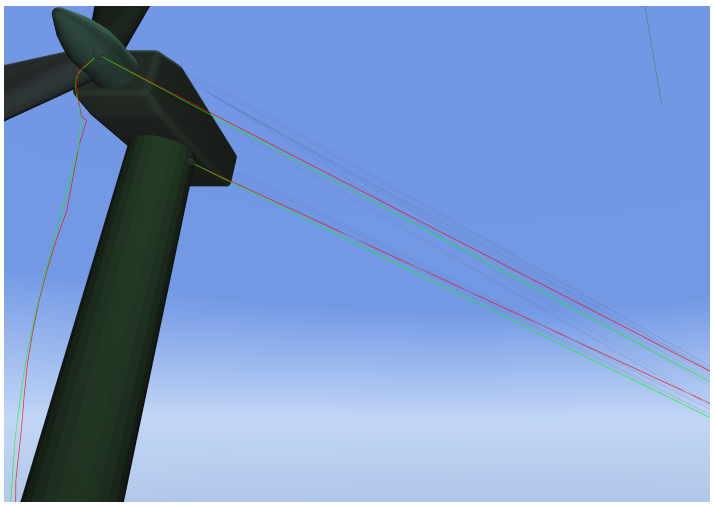

Figure 10. Path between vp 4 and vp 6. The approximated path is drawn in red and the geodesic path is drawn in green. The rotor is drawn as wire-frame to allow easier view of the path between vp 5 and 6.

path, a finite set of points can be selected on the edges of the configuration space and a solution sought. The solution should get nearer to optimal as the resolution and number of nodes increases. Methods for generating a configuration space exists and will generally have 6 degrees of freedom for a vehicle such as a helicopter. In this paper we reduce the complexity of the problem by replacing the geometric model of the vehicle with a sphere and using the convex hull of each work space. This result in a continuous differentiable configuration space. We propose a new method for generating the visibility graph from the configuration space, where the result of searching the graph will result in a path that gets closer to continuous differentiable as the resolution of the visibility graph improves. This means that the path should be easier to track as the resolution increases.

The test scenarios presented here have only been focused on off-line path planning. The approach described in this work tries to reuse static elements in the environment to reduce the number of links that have to be tested for intersection. Re-planning searches can be done in O(n) in worst case, where n is the number of nodes, for a visibility graph. It could be relevant to test under which conditions a visibility graph could be used for online calculations.

V. Conclusion

A method for generating a 3D visibility graph that can be searched to provide a path was presented. The applied method is based on first constructing continuous differentiable configuration space obstacles from geometric workspace obstacle, then generating a local visibility graph for each configuration space obstacle that approximates their surface without intersecting it. It was shown that a usable path can be found by searching the visibility graph using a graph search algorithm and that the found solution is close to the optimal for the given test scenario.

References

[1]Shim, D. H, *Autonomous exploration in unknown urban environments for unmanned aerial vehicles*, Technical report, University of California, Berkeley, 2005.

[2]Office of the Secretary of Defense, *Unmanned aircraft systems roadmap 2005-2030*, August 2005.

[3]M. Pachter and P. R. Chandler, *Challenges of Autonomous Control*, IEEE Control Systems Magazine, vol. 18, no. 4, August 1998.

[4]L. E. Dubins, *On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents*, Amer. J. Math, vol. 79, 1957.

[5]E. P. Anderson, R. W. Beard, and T. W. McLain, *Real-Time Dynamic Trajectory Smoothing for Unmanned Air Vehicles*, Ieee Transactions on Control Systems Technology, vol. 13, no. 3, May 2005.

[6]E. P. Anderson and R. W. Beard, *An Algorithmic Implementation of Constrained Extremal Control for UAVs*, AIAA Guidance, Navigation, and Control Conference and Exhibit, Monterey, California, 2002, AIAA 2002-4470.

[7]S. M. LaValle, *Planning Algorithms*, Cambridge University Press, 2006.

[8]G. Ambrosino, M. Ariola, U. Ciniglio, F. Corraro, A. Pironti and M. Virgilio, *Algorithms for 3D UAV Path Generation and Tracking*, 45th IEEE Conference on Decision & Control, 2006.

[9]T. I. Fossen, M. Breivik, and R. Skjetne, *Line-of-Sight Path Following of Underactuated Marine Craft*, Proceedings of the 6th IFAC MCMC, Girona, Spain, 2003, pp. 244-249.

[10]M. Bisgaard, *Modelling, estimation and control of helicopter slung load system*, PhD Thesis, Aalborg University, Department of Electronic Engineering, 2007.

[11]T. Lozano-Perez, M. A. Wesley, *An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles*, Communications of the ACM, 22 (1979), pp. 560-570.

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

[13]M. N. Bygi, M. Ghodsi. *3D Visibility Graph.*

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

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

[16]C. B. Barber, D. P. Dobkin, H. T. Huhdanpaa, *The quickhull algorithm for convex hulls*, ACM Trans. Mathematical Software 22, 469-483, 1996.

[17]R. Seidel, *Output-size sensitive algorithms for constructive problems in computational geometry*, Ph.D. thesis, Faculty of the Graduate School, Ithaca, NY, USA (1987).

[18]M. Kallay, *The complexity of incremental convex hull algorithms in Rd*, Inf. Process. Lett. 19 (4) (1984) 197. 46

[19]D. Chand, S. Kapur, *An algorithm for convex polytopes*, J. ACM 17 (1) (1970) 78-86.

[20]F. Preparata, S. Hong, *Convex hulls of finite sets of points in two and three dimensions*, Commun. ACM 20 (2) (1977) 87-93.

[21]R. Graham, *An effcient algorithm for determining the convex hull of a finite planar set*, Inf. Process. Lett. 1 (4) (1972) 132-133.

[22]J. O'Rourke, *Computational Geometry in C*, 2nd edition, Cambridge University Press, New York, NY, USA, 1998.

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

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

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

[26]T. Moller, B. Trumbore, *Fast, Minimum Storage Ray-Triangle Intersection*, J. Graphics Tools 2(1), 21-28 (1997).

[27]Z. C Marton, D. Pangercic, N. Blodow, J. Kleinehellefort, M. Beetz, *General 3D Modelling of Novel Objects from a Single View*, IEEE International Conference on Intelligent Robots and Systems, 2010.

[28]O. Chum, *Two-View Geometry Estimation by Random Sample and Consensus*. PhD Thesis, (2005).

[29]S. R. Buss, J. Fillmore, *Spherical Averages and Applications to Spherical Splines and Interpolation.*, ACM Transactions on Graphics 20, 2001, 95-126.

[30]E. Haines. *point in polygon strategies*, Graphics gems iv, 1994.