AIAA Guidance, Navigation, and Control Conference and Exhibit
16 - 19 August 2004, Providence, Rhode Island

AIAA 2004-5144

Three Dimensional Receding Horizon Control for UAVs

Yoshiaki Kuwata* and Jonathan How†

Massachusetts Institute of Technology

August, 2004

This paper presents a receding horizon controller (RHC) that can be used to design trajectories for an aerial vehicle flying through a three dimensional terrain with obstacles and no-fly zones. To avoid exposure to threats, the paths are chosen to stay as close to the terrain as possible, but the vehicle can choose to pop-up over the obstacles if necessary. The approach is similar to our previous two-dimensional algorithms that construct a coarse cost map to provide approximate paths from a sparse set of nodes to the goal and then use Mixed-integer Linear Programming (MILP) optimization to design a detailed trajectory. The main contribution of this paper is to extend this approach to 3D, in particular providing a new algorithm for connecting the cost map and the detailed path in the MILP. This connection is done by introducing a new cost-to-go function that includes an altitude penalty and accounts for the vehicle dynamics. Initial guess for MILP RHC is constructed from the previous solution and is shown to reduce the solution time. Several simulation results are presented to show that the path planning algorithm yields good overall performance and is computationally tractable in a complex environment.

Keywords Receding Horizon Control, Trajectory Optimization

I. Introduction

With the enhancing capability of Unmanned Aerial Vehicles (UAVs), their operation areas are being expanded to very complicated environments (e.g. urban) that have complex terrain.[1,2] In these environments the vehicles can go over or around the obstacles or no-fly zones, so path planning in three dimensions (3D) is a key technology to achieve the mission goals. In the past, vehicle guidance algorithms that avoid obstacles or other vehicles have been well studied in the areas of air traffic control, ground vehicles, and even UAVs. However, they typically assume the vehicle remains in a horizontal plane so that the path planning is two dimensional.[3–5] This paper presents a new guidance method for vehicles flying in 3D environments to reach the target in minimum time. This method builds on the extensive literature in the fields of computational geometry and robotics on shortest path problems on 2D polygons, 3D surfaces, and 3D spaces.[6–8] Similar to previous results in Ref.[9,10] our approach combines these shortest path algorithms with path planning techniques that use the vehicle dynamics to produce kinodynamically feasible trajectories that guide the vehicle to the goal.

The detailed trajectory optimization is conducted using Mixed-integer Linear Programming (MILP), which is well suited to trajectory planning because it can directly incorporate logical constraints such as obstacle avoidance and waypoint selection and because it provides an optimization framework that can account for basic dynamic constraints such as turn limitations and maximum rate of climb. The *receding horizon* approach (RH-MILP) enables us to exploit the power of this MILP formulation in a computationally tractable algorithm.[9,10] It solves a MILP for a detailed trajectory that only extends part of the way towards the goal. The remainder of the maneuver is represented by a cost-to-go function using path approximations.

* Research Assistant, MIT Dept. of Aeronautics and Astronautics, **kuwata@mit.edu**
† Associate Professor, MIT Dept. of Aeronautics and Astronautics, **jhow@mit.edu**.
Room 33-328, 77 Mass. Ave., Cambridge, MA 02139.

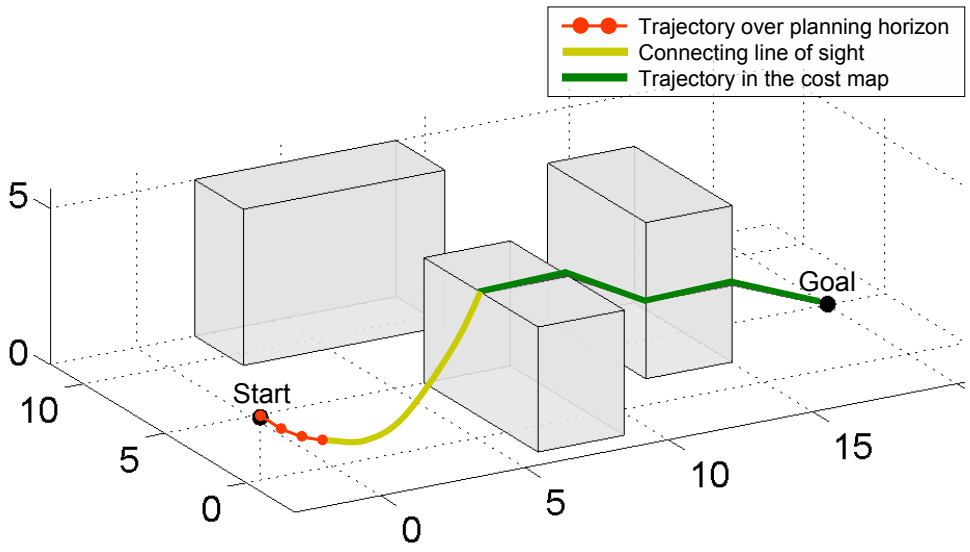


Figure 1: Schematic showing the three different resolution levels used in the RH-MILP approach to trajectory optimization.

Previous work on RH-MILP was limited to 2D environment, and presented heuristics that used straight line paths to estimate the cost-to-go from the plan's end point to the goal.[9] With some modifications to the formulation Ref.[10,11] proved that this RH-MILP approach is stable and that the vehicles reach the goal in finite time. These extensions compensated for the differences between the straight-line approximations in the cost-to-go calculation and the dynamically feasible paths that would be followed by the aircraft. Further extensions are required if the vehicles are to fly close to the surface of a 3D terrain in order to avoid threats such as radars. In these cases, the vertical vehicle maneuvers (*e.g.,* descend, climb up) have a significant effect on the overall trajectory, and a new cost-to-go function is needed to better estimate the future vehicle maneuvers.

This paper extends this approach to 3D, in particular providing a new algorithm for connecting the cost map and the detailed path in the MILP. This connection is achieved by introducing a new cost-to-go function that includes an altitude penalty and accounts for the vehicle dynamics. Several simulation results are presented to show that the path planning algorithm yields good overall performance in a complex environment. Also, an algorithm to provide starting values with MILP is presented in Section V, and is shown to reduce the solution time.

II. Algorithm Overview

Figure 1 shows the three resolution levels used in RH-MILP approach. In the near term, the MILP optimization solves for a detailed trajectory that extends from the current position towards the goal, but does not necessarily reach it. The line with bullets in Figure 1 shows this segment which is called planning horizon. In the far term, approximate trajectories from vertices on the obstacles to the goal are solved by a graph search and stored in the cost map. They are used to account for decisions beyond the planning horizon by estimating the time to reach the goal from the plan's end point. These two trajectories are then connected through the cost-to-go function in the receding horizon controller (RHC). The detailed trajectory is re-optimized on-line by the RHC while the vehicle executes the previous plan. The approximate trajectories are also updated on-line as knowledge of the environment changes. Splitting the problem into these different levels of resolution significantly reduces the computational effort to solve for the detailed vehicle trajectory while ensuring that the future decisions are (at least approximately) taken into account.

The proposed algorithm consists of two phases: the cost map construction (Section III) and the detailed trajectory optimization (Section IV). In the cost map construction phase, the environment is first mapped to a visibility graph consisting of nodes and arcs (Subsection A). The nodes represent candidate trajectory

points that the vehicle will fly through, and each arc connecting two nodes represents an approximate trajectory between them. The visibility between each pair of nodes needs to be ensured so that the arc connecting them is collision free and flyable. Subsection A presents a Linear Program (LP) that can be used to check the visibility. This new LP formulation is very flexible and can be used on-line for complex environments. The next step is to compute the shortest paths from the coarse grid of nodes to the goal using Dijkstra's algorithm. The results are then stored as a cost map (Subsection C). The accuracy of the path approximation depends on the node location. However, finding the exact shortest path in 3D environments is shown to be computationally intractable,[12] even without the vehicle dynamics, and Section III approximates the shortest paths by introducing nodes on obstacle edges.

In the detailed trajectory optimization phase, MILP is used to formulate the overall problem. First, Subsection A extends the vehicle dynamics to 3D. Subsection B presents a new cost-to-go function that is required to connect the detailed trajectory provided by MILP and the cost map produced by the graph search. Note that the limited set of nodes in the visibility graph allows the MILP to select an approximate routes from a coarse set of choices, significantly reducing the computation load.

III. Coarse Cost Map

This section presents a cost map that can be used to find approximate paths from a set of nodes to the goal. The formulation below assumes that each obstacle has a convex shape. Non-convex obstacles can be easily formed by having multiple convex obstacles intersect with each other. In two-dimensional cases, the corners of the obstacles together with the start and the goal points form a set of nodes in the visibility graph. In the three-dimensional case, however, shortest paths rarely visit obstacle corners.[6] This paper approximates the candidate nodes of shortest paths with obstacle corners on the ground ($z = 0$) and a middle point of each edge above ground-level. More vertices can be introduced on each obstacle edge, but the computation load both in the cost map construction phase and in the detailed trajectory design phase grows rapidly with small improvements in the accuracy.[6]

A. Visibility Graph

This section presents the visibility graph construction in an LP form. Our previous approaches assumed that the obstacles are 2D rectangles,[9,11] but this is not scalable to 3D environments. The new formulation presented in this section is much simpler and it can handle any convex obstacles. It also allows fast computation using commercially available software such as CPLEX.

In 3D environments, each convex obstacle is a polygon, as shown in Figure 1. Let π_k denote the k^{th} polygon, then,

$$\pi_k : A_k \boldsymbol{r} + \boldsymbol{b}_k \leq \boldsymbol{0} \tag{1}$$

where $\boldsymbol{r} = [x,\, y,\, z]^T$, and the row vectors of the matrix $[A_k \,|\, \boldsymbol{b}_k]$ are linearly independent of each other. Polygon π_k blocks the visibility of two nodes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ if there exists a point $\boldsymbol{r}$ that satisfies Eq. (1) and the conditions:

$$\begin{aligned} \boldsymbol{r} &= \boldsymbol{x}_i + l\,(\boldsymbol{x}_j - \boldsymbol{x}_i) && (2)\\ 0 &\leq l \leq 1 && (3) \end{aligned}$$

As shown in Figure 2, Eqs. (2) and (3) ensure that the point $\boldsymbol{r}$ is on the line connecting the two nodes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, and Eq. (1) ensures $\boldsymbol{r}$ is inside the polygon π_k. Given this definition, the visibility between all the nodes for all the obstacles can be determined by solving the following LP.

$$\min_{\boldsymbol{r}_{ijk},\, c_{ijk}} \left(\sum_{i,j,k\,(i<j)} c_{ijk} \right) \tag{4}$$

subject to

$$\begin{aligned} A_k \boldsymbol{r}_{ijk} + \boldsymbol{b}_k &\leq c_{ijk}\,\boldsymbol{1} && (5)\\ c_{ijk} &\geq 0 && (6)\\ \boldsymbol{r}_{ijk} &= \boldsymbol{x}_i + l_{ijk}\,(\boldsymbol{x}_j - \boldsymbol{x}_i) && (7)\\ 0 &\leq l_{ijk} \leq 1 && (8) \end{aligned}$$

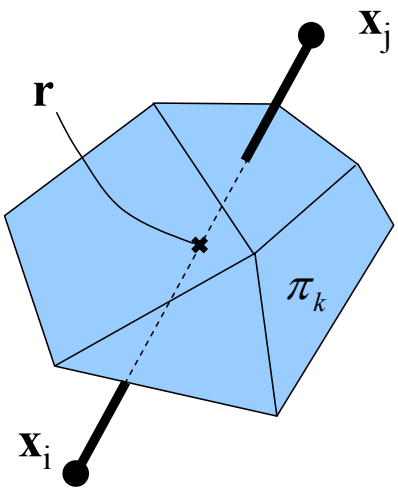


Figure 2: Thick line shows the arc connecting a pair of nodes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. The visibility between this pair is blocked by the obstacle π_k. The intersection point $\boldsymbol{r}$ is inside the polygon.

$$\begin{aligned} \boldsymbol{r}_{ijk} &= [x_{ijk},\, y_{ijk},\, z_{ijk}]^T \\ &\quad \forall\, i, j, k \quad (i < j) \end{aligned}$$

where the subscripts i and j represent the nodes in the visibility graph, and the subscript k represents the obstacles. If the visibility between a node pair (i, j) is obstructed by the k^{th} obstacle, there exists a point $\boldsymbol{r}_{ijk}$ such that $A_k \boldsymbol{r}_{ijk} + \boldsymbol{b}_k \leq 0$. Then, c_{ijk} is not constrained by Eq. (5), and Eqs. (4) and (6) make $c_{ijk} = 0$. If the visibility is not obstructed, then Eq. (5) forces c_{ijk} to be positive.

Based on this discussion, the solution of the LP, c_{ijk}, can be used to determine the visibility between each pair of nodes (i, j). The nodes (i, j) are mutually visible if

$$c_{ijk} > 0, \quad \forall\, k \tag{9}$$

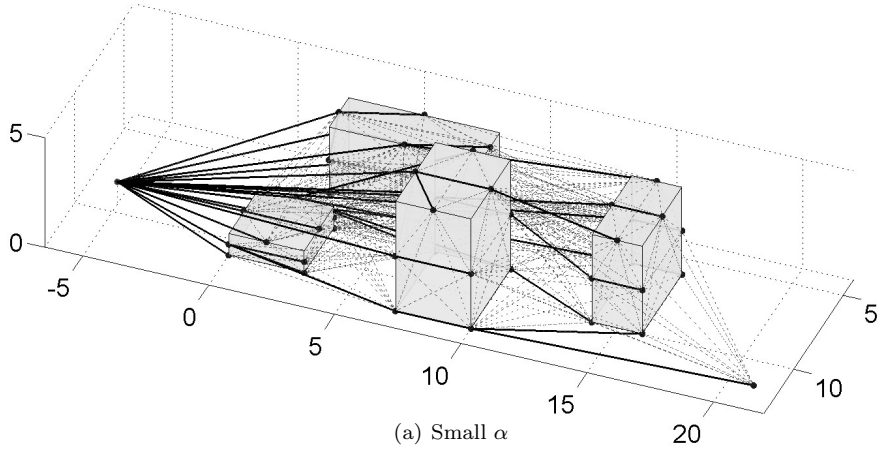
If Eq. (9) is not satisfied, then at least one obstacle obstructs the visibility, as shown in Figure 2. Note that the LP solution includes the visibility information on all pairs of nodes for all the obstacles which allows for a fast incremental update of the visibility graph when the environment changes.[13,14]

B. Arc Lengths

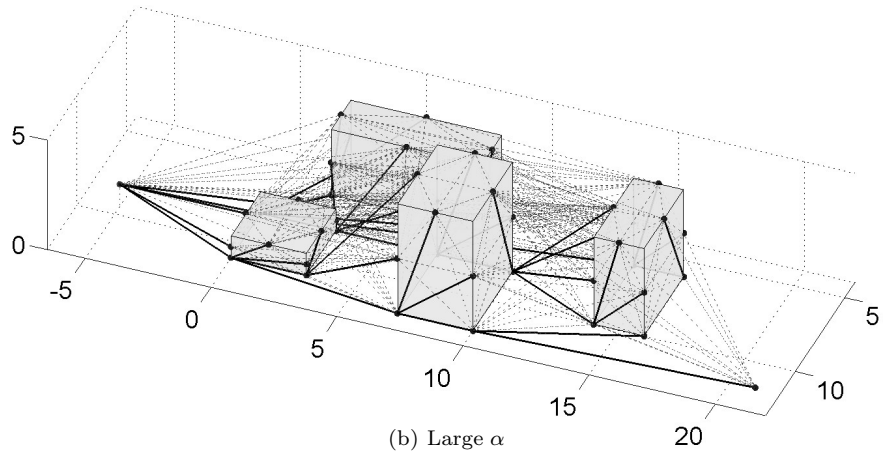
Given the visibility between the two nodes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, the next step is to calculate the arc cost D_{ij} between the two nodes, which represents the length and the threat exposure of the path connecting them. To avoid threats and radar detection, it is assumed that the vehicle would like to stay as low as possible. This objective is captured by penalizing the altitude of the path with a weight α. Thus, D_{ij} includes the straight line (Euclidean) distance between the nodes and the path integral of the altitude along the straight line connecting the nodes.

$$D_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 \left(1 + \alpha \frac{z_i + z_j}{2}\right) \tag{10}$$

This section examines candidate trajectories for a far future. Thus, the straight line trajectories are used simply to obtain the distance and identify the approximate threat level associated with it.



(a) Small α



(b) Large α

Figure 3: Shortest path from each node (•) to the goal in the left.

C. Cost Map

Once the visibility graph is constructed, Dijkstra's algorithm is used to find the shortest path from each node to the goal in the visibility graph.[9] Note that the "shortest" path here is determined based on the arc cost and not necessarily the Euclidean distance. Figure 3 illustrates the effect of the altitude penalty α on the shortest path. The dashed lines show the visibility graph, and the thick lines show the shortest path from each node to the goal. With a small penalty on the altitude (Figure (a)), direct connections from the goal to nodes are always shortest paths. However, with a large penalty on the altitude (Figure (b)), the shortest paths tend to consist of arcs along ground level.

The output of the Dijkstra's algorithm contains the cost C_i from each node i to the goal and successors of each node on the way to the goal. This output is stored as a cost map, and gives an approximate cost-to-go at each node in the MILP optimization, as discussed in the next section.

IV. Detailed Plan

A. Vehicle Model

The vehicle model presented in this section captures the key characteristics of the aircraft dynamics in the MILP framework.[15,16] This is done by imposing constraints on the maximum and minimum speed, maximum turn rate, the maximum rate of climb, and the maximum rate of descent. The linearized vehicle dynamics in a discretized form can be written as

$$\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{v} \end{bmatrix}_{k+1} = A \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{v} \end{bmatrix}_{k} + B\,\boldsymbol{a}_k \tag{11}$$

$$
\begin{aligned}
\boldsymbol{x} &= \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad \boldsymbol{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}, \quad \boldsymbol{a} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \\
A &= \left[\begin{array}{c|c} I_3 & \Delta t \cdot I_3 \\ \hline O_3 & I_3 \end{array}\right], \quad B = \left[\begin{array}{c} \frac{(\Delta t)^2}{2} I_3 \\ \hline I_3 \end{array}\right]
\end{aligned}
$$

where the subscript k represents the discrete time-step, I_3 represents an identity matrix of size 3×3, and O_3 is a zero matrix of size 3×3. Vectors $\boldsymbol{x}, \boldsymbol{v}$, and $\boldsymbol{a}$ respectively represent position, velocity, and acceleration input in the inertia frame. The following constraints limit the magnitude of the acceleration and velocity vectors, which in turn limits the maximum turning rate and the maximum pitching rate, provided that the optimization favors minimum time solutions.[16]

$$
\begin{aligned}
L_2(\boldsymbol{a}) &\leq a_{\max} \qquad (12) \\
L_2(\boldsymbol{v}) &\leq v_{\max} \qquad (13)
\end{aligned}
$$

where $L_2(\boldsymbol{r})$ approximates the upper bound of the 2-norm of a vector $\boldsymbol{r}$. This approximation uses n unit vectors that are distributed in the 3D space

$$
\begin{aligned}
L_2(\boldsymbol{r}) &\geq \boldsymbol{r} \cdot \boldsymbol{i}_m, \qquad m = 1, \ldots, n \qquad (14) \\
\boldsymbol{r} &= [r_x,\ r_y,\ r_z]^T \\
\boldsymbol{i}_m &= \left[\ \sin\phi_m \cos\theta_m,\ \ \sin\phi_m \sin\theta_m,\ \ \cos\phi_m\ \right]^T
\end{aligned}
$$

Non-convex constraints on the minimum speed

$$
\begin{aligned}
v_x \cos\theta_m + v_y \sin\theta_m &\geq v_{\min} - 2 v_{\max} b_{\text{speed},m} \qquad m = 1, \ldots, n_v \qquad (15) \\
\sum_{m=1}^{n_v} b_{\text{speed},m} &\geq 1 \qquad (16)
\end{aligned}
$$

prevent the vehicle from stalling. Constraints on the maximum rate of climb and descent are written as

$$
v_{z,\min} \leq v_z \leq v_{z,\max} \qquad (17)
$$

Finally, a list of waypoint commands is sent to the vehicle which is augmented with a waypoint tracking controller. Other vehicle models are currently under investigation that better capture more detailed vehicle dynamics.[17]

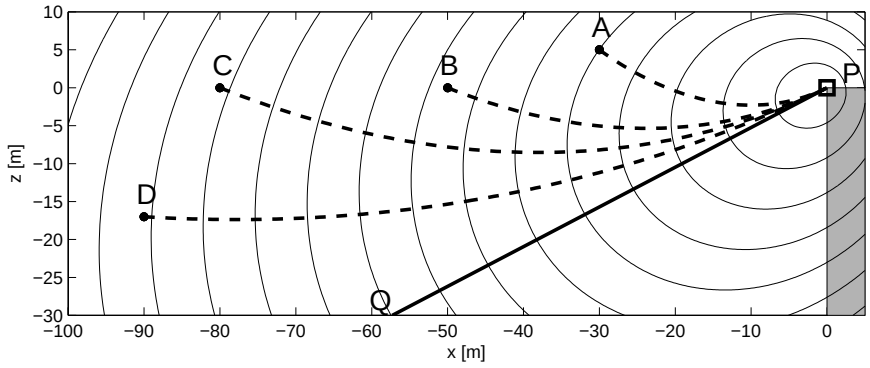
B. Cost-To-Go Function

The RHC represents the plan beyond the planning horizon by evaluating a cost-to-go function at the terminal state. The cost-to-go function in the previous work used straight lines from the terminal state to the selected cost point because it gave a good approximation of the optimal trajectory.[9] However, this is not the case in 3D environments, and the terminal penalty needs to be revised to account for the change in the altitude. In the cost map construction phase, Eq. (10) takes the line integral of the altitude along the straight line to approximate the altitude penalty in the future trajectory. In the detailed trajectory phase, the detailed altitude profile of the vehicle is obtained over the short horizon. The new cost-to-go function presented here allows us to connect these two trajectories while accounting for the altitude penalty and the vehicle dynamics.

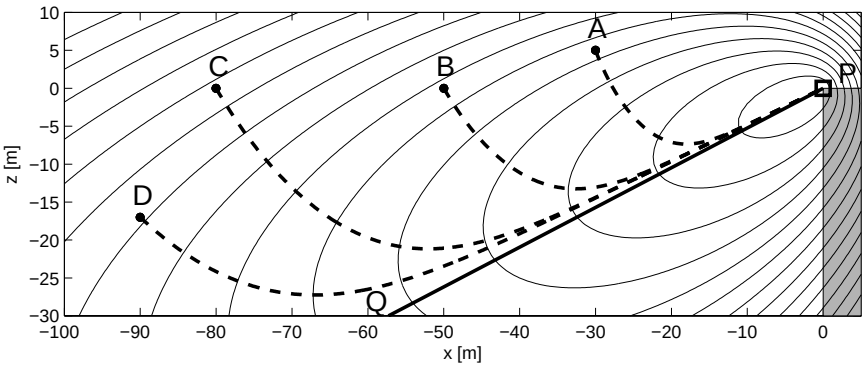
In order to simplify the presentation, the analysis in this subsection only examines the motion in the x-z plane. The final result in Subsection C accounts for the full 3D motion. Let $\boldsymbol{x}_{\text{vis}} = [x_{\text{vis}}, z_{\text{vis}}]$ denote a "visible" point that the vehicle is aiming for. Then, cost-to-go function used in this paper can be written as

$$
\begin{aligned}
F(x,z) &= \sqrt{(x - x_{\text{vis}})^2 + (z - z_{\text{vis}})^2} + \alpha z - \beta (x_{\text{vis}} - x) \qquad (18) \\
&\alpha > 0,\ \beta > 0
\end{aligned}
$$

where the first term represents the Euclidean distance between the point $[x, z]$ and the visible point $\boldsymbol{x}_{\text{vis}}$, the second and the third term separately penalize vertical and horizontal motion. In order to illustrate the



(a) $p = 0.25$



(b) $p = 0.75$

Figure 4: Contour maps of the cost-to-go function in x-z plane. Solid line represents the contour around the visible point P. Dashed lines show the steepest descent lines from four points (A, B, C, D) to the visible point. In this example, $\boldsymbol{x}_{\text{vis}} = [0, 0]^T$.

effect of this cost-to-go function, Figure 4 shows a contour map of the cost-to-go function around the visible point P, which is marked with □. The dashed lines in Figure 4 show the steepest descent lines from four arbitrary points (A, B, C, D) to the visible point. By minimizing the cost-to-go function, the vehicle lowers its altitude to reduce the altitude penalty when the vehicle is far from the visible point and its altitude is high. As it moves closer to the visible point, the trajectory converges to the limiting line PQ.

The second and the third term $\alpha z - \beta(x_{\text{vis}} - x)$ in Eq. (18) determine the angle of this line PQ. It can be shown geometrically that the major axis of the ellipse in Eq. (18) forms an angle $\gamma_{\max}$ with x axis, where

$$\tan \gamma_{\max} = \frac{\alpha}{\beta} \tag{19}$$

This angle $\gamma_{\max}$ represents the maximum path angle of the vehicle, and once the vehicle crosses the line PQ, it cannot avoid colliding with the gray obstacle on the right. However, the plots of the steepest descent lines show that by minimizing the cost-to-go function in Eq. (18), the vehicle trajectory will not cross the line PQ.

In order for the cost-to-go function to navigate the vehicle to the visible point P, it is required that

$$p \equiv \alpha^2 + \beta^2 < 1 \tag{20}$$

Finally, the coefficients α and β in Eq. (18) can be obtained from the following equations, given the maximum path angle $\gamma_{\max}$ and a parameter p.

$$\alpha = \frac{\gamma_{\max}\sqrt{p}}{\sqrt{{\gamma_{\max}}^2 + 1}} \tag{21}$$

$$\beta = \frac{\sqrt{p}}{\sqrt{{\gamma_{\max}}^2 + 1}} \tag{22}$$

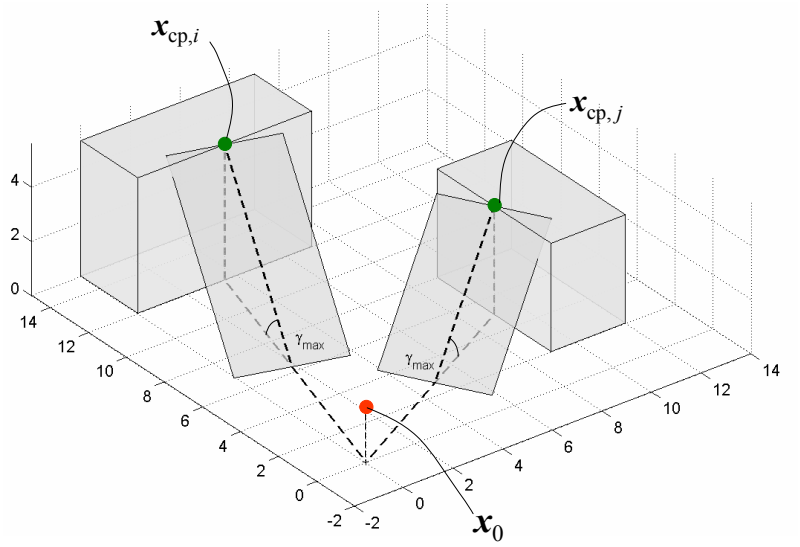


Figure 5: Dashed lines show path approximation by the cost-to-go function in 3D. Each plane is formed with two axes of the contour ellipsoid.

Choosing a larger p produces a flatter ellipse, and hence tighter trajectories. Figures 4(a) and (b) compare two contours with the same $\gamma_{\max}$ but different p. The dashed lines in the Figure 4(b) have tighter descent trajectories. Note that although the cost-to-go function includes the ascending vehicle dynamics only, the combination of the vehicle dynamics in Subsection A and the cost-to-go function in this subsection produces a dynamically feasible trajectory over the planning horizon and kinodynamically feasible rate-of-climb commands towards the visible point.

C. MILP RHC

In the detailed trajectory optimization phase, MILP uses a binary variable $\boldsymbol{b}_{\rm vis}$ to select one visible point $\boldsymbol{x}_{\rm vis}$ from a list of cost points from which the cost-to-go is known. Let $\boldsymbol{x}_{{\rm cp},i}$ denote the $i^{\rm th}$ cost point and $i = 1, \ldots, n_{\rm cp}$ where $n_{\rm cp}$ is a number of cost points. Then,

$$\boldsymbol{x}_{\rm vis} = \sum_{i=1}^{n_{\rm cp}} b_{{\rm vis},i}\, \boldsymbol{x}_{{\rm cp},i} \tag{23}$$

$$1 = \sum_{i=1}^{n_{\rm cp}} b_{{\rm vis},i} \tag{24}$$

In order to connect the detailed 3D trajectory to the selected cost point, Eq. (18) is extended here to 3D

$$\begin{aligned} F_i(x,y,z) &= \sqrt{(x-x_{{\rm cp},i})^2 + (y-y_{{\rm cp},i})^2 + (z-z_{{\rm cp},i})^2} \\ &\quad + \alpha z - \beta \left\| \begin{bmatrix} x_{{\rm cp},i} - x \\ y_{{\rm cp},i} - y \end{bmatrix} \right\|_2 \qquad (i = 1, \ldots, n_{\rm cp}) \end{aligned} \tag{25}$$

RHC optimizes the vehicle trajectory over a short planning horizon of n_p steps, executes only the first n_e steps of the control input, and starts the next optimization from the state that the vehicle will reach. Each optimization produces a detailed, but short, trajectory, which allows us to assume that the trajectory point $\boldsymbol{x}$ lies close to a vertical plane passing through a cost point $\boldsymbol{x}_{{\rm cp},i}$ and the initial position $\boldsymbol{x}_0$. In this case, we can approximate

$$\left\| \begin{bmatrix} x_{{\rm cp},i} - x \\ y_{{\rm cp},i} - y \end{bmatrix} \right\|_2 \simeq \left\| \begin{bmatrix} x_{{\rm cp},i} - x_0 \\ y_{{\rm cp},i} - y_0 \end{bmatrix} \right\|_2 - \left\| \begin{bmatrix} x - x_0 \\ y - y_0 \end{bmatrix} \right\|_2 \tag{26}$$

If $\boldsymbol{x}$ lies on the vertical plane passing through $\boldsymbol{x}_{\mathrm{cp},i}$ and $\boldsymbol{x}_0$,

$$\left\| \begin{bmatrix} x-x_0 \\ y-y_0 \end{bmatrix} \right\|_2 = (x-x_0)\cos\theta_i + (y-y_0)\sin\theta_i \tag{27}$$

$$\tan\theta_i = \frac{y_{\mathrm{cp},i}-y_0}{x_{\mathrm{cp},i}-x_0} \tag{28}$$

where θ_i represents the direction of a vector from the initial position to the i^{th} cost point, projected onto the x-y plane. Note that this θ_i's are calculated prior to MILP, and are given as parameters to MILP. Let d_i denote the Euclidean distance between $\boldsymbol{x}_0$ and $\boldsymbol{x}_{\mathrm{cp},i}$. Then,

$$\begin{aligned} F_i(x,y,z) &\simeq \sqrt{(x-x_{\mathrm{cp},i})^2+(y-y_{\mathrm{cp},i})^2+(z-z_{\mathrm{cp},i})^2} \\ &\quad + \alpha z + \beta\left\{ \left\| \begin{bmatrix} x-x_0 \\ y-y_0 \end{bmatrix} \right\|_2 - \left\| \begin{bmatrix} x_{\mathrm{cp},i}-x_0 \\ y_{\mathrm{cp},i}-y_0 \end{bmatrix} \right\|_2 \right\} \\ &\simeq \sqrt{(x-x_{\mathrm{cp},i})^2+(y-y_{\mathrm{cp},i})^2+(z-z_{\mathrm{cp},i})^2} \\ &\quad + \alpha z + \beta\Big\{ (x-x_0)\cos\theta_i + (y-y_0)\sin\theta_i - d_i \Big\} \end{aligned} \tag{29}$$

The third term $\beta\{\cdot\}$ in Eq. (29) is equivalent to the third term in Eq. (18); it evaluates the horizontal distance from the point $\boldsymbol{x}$ to the selected cost point. For each cost point, the contour of Eq. (29) is ellipsoid, and its major axis makes an angle $\gamma_{\max}$ with the ground surface $z=0$, as shown in Figure 5. Note that this axis is equivalent to the line PQ in Figure 4.

This cost-to-go function F_i must also be expressed in a MILP form. The first term in Eq. (29) represents the two-norm of a vector, which can be approximated using a set of distributed unit vectors, as shown in Eq. (14). The third term $\beta\{\cdot\}$ can be obtained by minimizing βJ_{h}, where

$$J_{\mathrm{h}}(x,y) = \sum_{i=1}^{n_{\mathrm{cp}}} l_i \; - \sum_{i=1}^{n_{\mathrm{cp}}} b_{\mathrm{vis},i}\; d_i \tag{30}$$

with

$$l_i \geq (x-x_0)\cos\theta_i + (y-y_0)\sin\theta_i - n_p v\Delta t\,(1-b_{\mathrm{vis},i}) \tag{31}$$

$$l_i \geq 0 \tag{32}$$

$$(i=1,\ldots,n_{\mathrm{cp}})$$

If the i^{th} cost point is not selected, $b_{\mathrm{vis},i}=0$, and Eq. (31) is relaxed because the sum of the first two terms expresses the distance travelled in the direction of the i^{th} cost point, which is always smaller than the planning horizon length $n_p v\Delta t$. Minimizing J_{h} forces all the l_i's to equal zero except for the one associated with the cost point that is selected ($b_{\mathrm{vis},i}=1$). In particular, if the i^{th} cost point is selected, then

$$\min J_{\mathrm{h}} = \beta\Big\{ (x-x_0)\cos\theta_i + (y-y_0)\sin\theta_i - d_i \Big\}$$

as required.

The cost-to-go function connecting the final state $\boldsymbol{x}_{n_p}$ to each cost point has the global minimum at the cost point. This can be interpreted as a potential function surrounding each cost point. The decision variable $\boldsymbol{b}_{\mathrm{vis}}$ of the RHC allows the in-flight selection of the potential field. Path planning techniques using a potential function usually have difficulty handling local minima, but the dynamic mode switching by $\boldsymbol{b}_{\mathrm{vis}}$ avoids this issue.

Kinematic constraints including obstacle avoidance and the ground plane can be expressed in MILP using a binary variable $\boldsymbol{b}_{\mathrm{obst}}$.[5] The constraints are applied to each trajectory point over the planning horizon. To ensure that the selected cost point $\boldsymbol{x}_{\mathrm{vis}}$ is "visible" from the terminal point $\boldsymbol{x}_{n_p}$, several sample points are placed on the line connecting these two points, and kinematic constraints are applied also to them. For each point $\boldsymbol{x}=[x,y,z]^T$ and each rectangular column shaped obstacle defined by two corners $[x_{\mathrm{low}},y_{\mathrm{low}},z_{\mathrm{low}}]^T$

and $[x_{\text{high}}, y_{\text{high}}, z_{\text{high}}]^T$, the avoidance constraints can be expressed as

$$\left.\begin{aligned} x &\le x_{\text{low}} + M\, b_{\text{obst},1} \\ y &\le y_{\text{low}} + M\, b_{\text{obst},2} \\ z &\le z_{\text{low}} + M\, b_{\text{obst},3} \\ x &\ge x_{\text{high}} - M\, b_{\text{obst},4} \\ y &\ge y_{\text{high}} - M\, b_{\text{obst},5} \\ z &\ge z_{\text{high}} - M\, b_{\text{obst},6} \end{aligned}\right\} \tag{33}$$

$$z \ge 0 \tag{34}$$

$$\sum_{i=1}^{6} b_{\text{obst},i} \le 5 \tag{35}$$

where M is a large number to relax the constraints in Eq. (33). The logical constraint Eq. (35) requires at least one constraint in Eq. (33) be active.

The RHC minimizes the sum of the state penalty over the planning horizon and the terminal penalty evaluated at the final state $\boldsymbol{x}_{n_p}$. The overall objective function J is then the sum of four terms

$$\min J = \min \left\{ \sum_{k=1}^{n_p} \left(\|\boldsymbol{x}_{\text{vis}} - \boldsymbol{x}_k\|_2 + \alpha z_k + \beta J_{\text{h}}(x_k, y_k) \right) + \sum_{i=1}^{n_{\text{cp}}} b_{\text{vis},i}\, C_i \right\} \tag{36}$$

The first term penalize the altitude over the planning horizon. The second term measures the distance from the terminal point to the selected cost point. The third term, together with the second term, generate a cost-to-go function from the terminal state to the selected cost point, as discussed in Subsection B. The last term represents the cost-to-go from the selected cost point to the goal, and this value is given by the cost map, as discussed in Section III.

The formulation presented in this paper used several approximations to significantly reduce the problem size of the complex trajectory optimization. The simulation results in Section VI demonstrate the validity of the approximations and show the overall MILP RHC has a good performance.

V. Initial Guess for MILP

In order to shorten the solution time of the MILP, an initial feasible solution can be provided with the solver. The integer feasible solution gives an upper bound on the optimal cost, which allows to prune some search trees in the branch-and-bound algorithm, shortening the search.[11] This paper examines 3D environments where only vertical obstacles exist. In such environments, one feasible solution is simply to fly up with its maximum acceleration.

RHC executes only the first n_e steps of the n_p step plan, and reoptimize from the state that will be reached. When constructing an initial guess, the decisions (e.g. visible point selection, obstacle avoidance) made in the previous solution could be used. An algorithm that construct an initial guess from the previous solution is summarized below.

- Cost point selection
  Choose the same visible point as the one in the previous solution.
- Input command
  For the first $(n_p - n_e)$ steps, reuse the last $(n_p - n_e)$ steps of the previous solution. For the rest, append $\boldsymbol{a} = [0, 0, a_z]^T$ where a_z is the maximum acceleration command that satisfies the constraints on the vehicle dynamics Eqs. (11) to (17).

This produces the vehicle states over the planning horizon and the glue that connects the detailed plan to the cost map. Based on this, finding binary variables for obstacle avoidance, target arrival, and minimum speed constraints is a deterministic operation and follows easily. The impact of the initial guess on the computation time is presented in the next section.

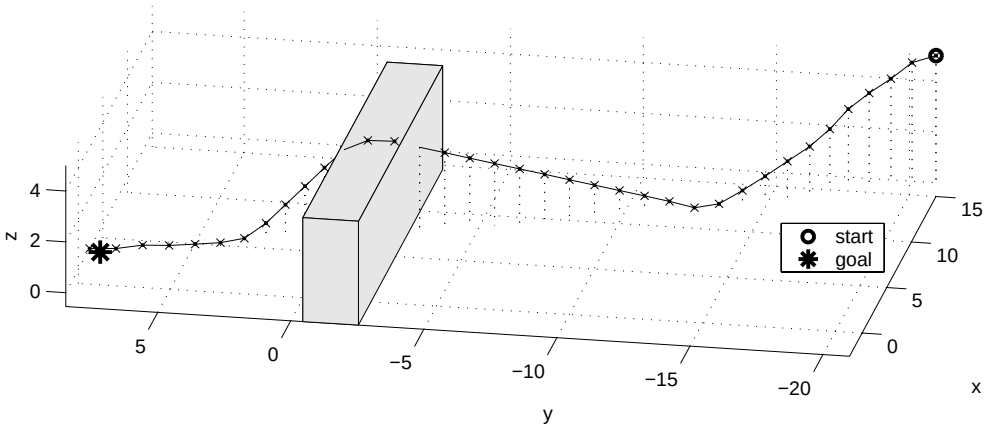


Figure 6: Trajectory generated by the RHC in a three dimensional environment. The vehicle starts at $\circ$, and the goal is marked with $*$.

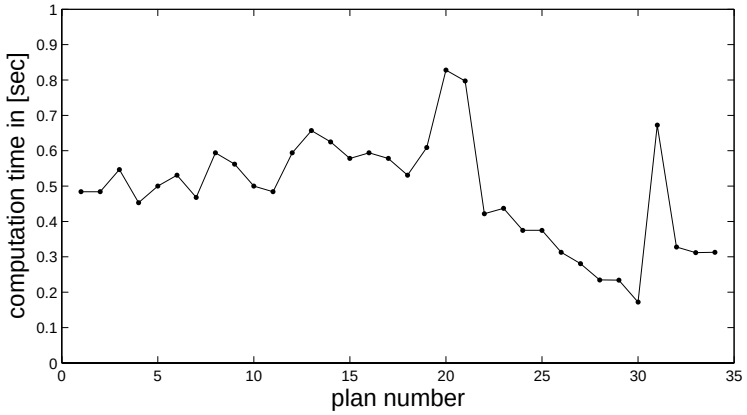


Figure 7: Computation time.

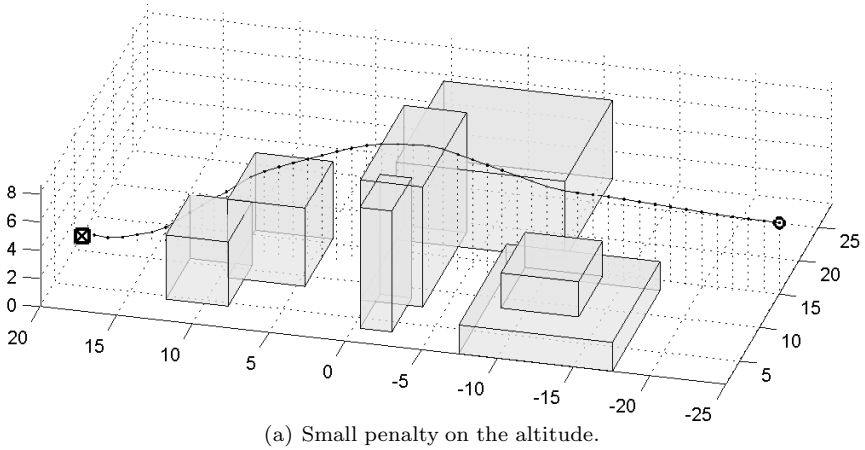
VI. Results

First, a simple problem has been solved using commercially available software CPLEX 9.0.[18] Figure 6 shows the resulting trajectory. The following parameters are used in the simulation.

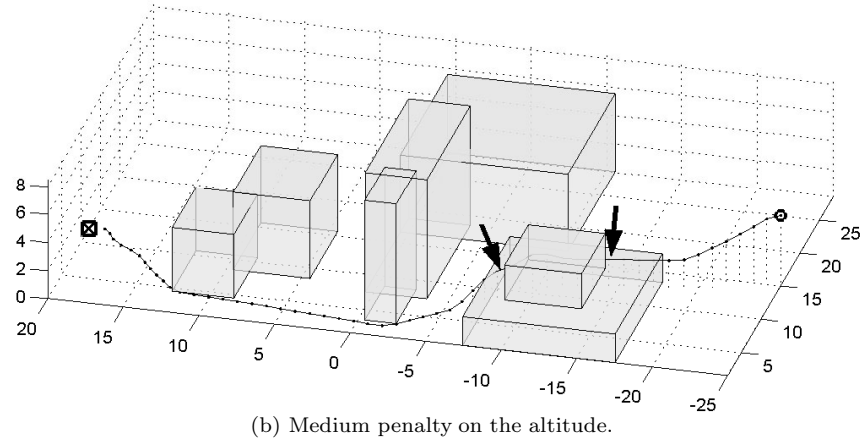
- $n_p = 4,\ n_e = 1$
- $\gamma_{\max} = 30\text{deg},\ p = 0.6$
- Number of nodes per obstacle = 12

The start point on the right is marked with the $\circ$, and the goal is on the left. To minimize the altitude, the vehicle descends from the start point, until it reaches ground level. Then as it approaches the obstacle, it starts a climb-up maneuver which is triggered by the cost-to-go function (see Figure 4). Note that the planning horizon is 4 steps in this example, and the RHC made different decisions (*e.g., descend, ascend*) while approaching the obstacle. Figure 7 shows the computation time for each MILP optimization on a PC (2GHz Pentium 4 CPU, with 1GB RAM).

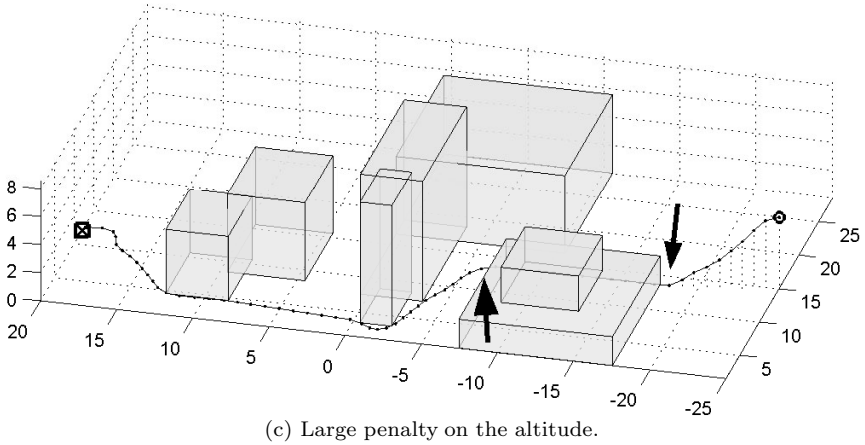
Figure 8 shows trajectories in a more complicated environment. Each figure corresponds to a different penalty on the altitude. If there is only a small penalty (Figure (a)), the vehicle flies over all of the obstacles, even including the tall ones. If projected onto the ground, the resultant trajectory is effectively a straight line connecting the start and the goal. With a larger altitude penalty (see Figure (b)) a very different trajectory



(a) Small penalty on the altitude.



(b) Medium penalty on the altitude.



(c) Large penalty on the altitude.

Figure 8: Trajectories generated by the RHC in a complex three dimensional environment. The vehicle starts at ○, and the goal is marked with ⊠.

Table 1: Comparison of computation time (seconds)

	W/O Initial Guess		With Optimal Solution		With Initial Guess	
	peak	ave.	peak	ave.	peak	ave.
Figure (a)	1.50	0.74	1.08	0.57	1.13	0.62
Figure (b)	2.55	0.82	1.34	0.59	1.30	0.67
Figure (c)	1.83	0.88	1.64	0.70	1.59	0.76

Table 2: Reduction of the computation time (%)

	With Optimal Solution		With Initial Guess	
	peak	ave.	peak	ave.
Figure (a)	28.1	23.4	25.0	16.3
Figure (b)	47.2	28.3	49.1	18.7
Figure (c)	10.3	20.3	12.9	14.3

is obtained. In this case the vehicle flies around most of the obstacles at a very low altitude. However, the two-story obstacle near the start of the trajectory (lower right of the figure) is directly in the way. The vehicle decides to fly over the first-story, skirting the outside of the second story. As the altitude penalty is increased further, Figure (c) shows that the vehicle goes around all the obstacles. The difference between Figure (b) and (c) is emphasized with arrows in the figures.

The true optimal solution is computationally intractable to obtain, but in the solutions presented here, the vehicle mostly keeps the maximum speed with the smooth trajectories, which indicates they are close to the optimal trajectory. Note that for this example the average computation time increases to $\sim$1 second because there are many choices to make in this complex and constrained environment.

Table 1 shows the CPLEX computation time in seconds for the scenarios presented in Figure 8. The first two columns respectively show the peak and average computation times without initial guess. The next two columns show the computation times when CPLEX is given the optimal solution as the MILP starting values. The last two columns show the computation times when the initial guess described in Section V is used. Table 2 shows the reduction of the computation time in percentage when initial guess values are used.

There is an overall reduction of 20–28% on average if the optimal solution is provided as the MILP starting values. The initial guess in Section V produced a slightly less improvement in the average computation time, but can still significantly reduce the worst case computation time.

VII. Conclusions

This paper presented a trajectory planning algorithm for the vehicle flying in 3D environments with obstacles and no-fly zones. The vehicle is required to fly close to the 3D surface to avoid exposure to threats while minimizing the time of arrival at the target. The proposed algorithm has two phases: the cost map construction and the detailed trajectory optimization. In the construction of a coarse cost map, linear programming has been applied to find the visibility graph, and the Dijkstra's algorithm is used to find the approximate shortest paths from each node to the goal. RHC designs a short but detailed trajectory using MILP while approximating the future maneuver by connecting the detailed trajectory to the coarse cost map. This is done by a new cost-to-go function which accounts for the vehicle dynamics and the altitude penalty beyond the planning horizon. Initial guess for the MILP RHC is constructed from the previous solution which further reduces the computation load. The simulation results showed that the overall approach is computationally tractable in complex 3D environments.

Acknowledgments

Research funded by AFOSR Grant # F49620-01-1-0453.

References

[1]Office of the Secretary of Defense, "Unmanned Aerial Vehicles Roadmap," Tech. rep., December 2002.

[2]Bay, J., "Heterogeneous Urban RSTA Team (HURT)," Tech. rep., DARPA/IXO, December 2003, www.darpa.mil/baa/baa04-05.htm.

[3]Bicchi, A. and Pallottino, L., "On Optimal Cooperative Conflict Resolution for Air Traffic Management Systems," *IEEE Trans. on Intelligent Transportation Systems*, Vol. 1-4, Dec 2000, pp. 221–231.

[4]Mao, Z. H., Feron, E., and Bilimoria, K., "Stability and Performance of Intersecting Aircraft Flows Under Decentralized Conflict Avoidance Rules," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 2, No. 2, 2001, pp. 101–109.

[5]Richards, A., Schouwenaars, T., How, J., and Feron, E., "Spacecraft Trajectory Planning With Collision and Plume Avoidance Using Mixed-Integer Linear Programming," *Journal of Guidance, Control and Dynamics*, Vol. 25, No. 4, Aug 2002, pp. 755–764.

[6]Gewali, L. P., Ntafos, S., and Tollis, I. G., "Path Planning in the Presence of Vertical Obstacles," *IEEE Transactions on Robotics and Automation*, Vol. 6, No. 3, June 1990, pp. 331–341.

[7]Aleksandrov, L., Lanthier, M., Maheshwari, A., and Sack, J.-R., "An epsilon Approximation Algorithm for Weighted Shortest Paths on Polyhedral Surfaces," *Proceedings of the 6th Scandinavian Workshop on Algorithm Theory. Lecture Notes in Computer Science*, Vol. 1432, 1998, pp. 11–22.

[8]Kanai, T. and Suzuki, H., "Approximate Shortest Path on Polyhedral Surface Based on Selective Refinement of the Discrete Graph and Its Applications ," *Geometric Modeling and Processing*, April 2000.

[9]Bellingham, J., Richards, A., and How, J., "Receding Horizon Control of Autonomous Aerial Vehicles," *Proceedings of the IEEE American Control Conference*, Anchorage, AK, May 2002, pp. 3741–3746.

[10]Kuwata, Y. and How, J., "Stable Trajectory Design for Highly Constrained Environments using Receding Horizon Control," *Proceedings of the IEEE American Control Conference*, Boston, MA, 2004.

[11]Bellingham, J., Kuwata, Y., and How, J., "Stable Receding Horizon Trajectory Control for Complex Environments," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Austin, TX, Aug 2003.

[12]Canny, J. and Reif, J., "New Lower Bound Techniques for Robot Mmotion Planning Problems," *28th Annual Symposium on IEEE Symposium on Foundations of Computer Science*, October 1987, pp. 49–60.

[13]Narvaez, P., Siu, K.-Y., and Tzeng, H.-Y., "New Dynamic Algorithms for Shortest Path Tree Computation," *IEEE/ACM Transactions on Networking*, Vol. 8, No. 6, December 2000, pp. 734–746.

[14]Koenig, S. and Likhachev, M., "Improved Fast Replanning for Robot Navigation in Unknown Terrain," *Proceedings of the IEEE International Conference on Robotics and Automation*, 2002.

[15]Schouwenaars, T., Moor, B. D., Feron, E., and How, J., "Mixed Integer Programming for Multi-Vehicle Path Planning," *Proceedings of the European Control Conference*, Porto, Portugal, September 2001.

[16]Richards, A. and How, J. P., "Aircraft Trajectory Planning With Collision Avoidance Using Mixed Integer Linear Programming," *Proceedings of the IEEE American Control Conference*, Anchorage, AK, May 2002, pp. 1936–1941.

[17]Schouwenaars, T., Feron, E., and How, J., "Hybrid Model for Receding Horizon Guidance of Agile Maneuvering Autonomous Rotorcraft," submitted to 16th IFAC Symposium on Automatic Control in Aerospace.

[18]ILOG, *ILOG CPLEX User's guide*, 1999.