Receding Horizon Control of Autonomous Aerial Vehicles [1]

John Bellingham [2], Arthur Richards [3], and Jonathan P. How [4]
Massachusetts Institute of Technology
Cambridge MA 02139

Abstract

This paper presents a new approach to trajectory optimization for autonomous fixed-wing aerial vehicles performing large-scale maneuvers. The main result is a planner which designs nearly minimum time planar trajectories to a goal, constrained by no-fly zones and the vehicle's maximum speed and turning rate. Mixed-Integer Linear Programming (MILP) is used for the optimization, and is well suited to trajectory optimization because it can incorporate logical constraints, such as no-fly zone avoidance, and continuous constraints, such as aircraft dynamics. MILP is applied over a receding planning horizon to reduce the computational effort of the planner and to incorporate feedback. In this approach, MILP is used to plan short trajectories that extend towards the goal, but do not necessarily reach it. The cost function accounts for decisions beyond the planning horizon by estimating the time to reach the goal from the plan's end point. This time is estimated by searching a graph representation of the environment. This approach is shown to avoid entrapment behind obstacles, to yield near-optimal performance when comparison with the minimum arrival time found using a fixed horizon controller is possible, and to work consistently on large trajectory optimization problems that are intractable for the fixed horizon controller.

1 Introduction

The capabilities and roles of Unmanned Aerial Vehicles (UAVs) are evolving, and require new concepts for their control. A significant aspect of this control problem is optimizing the long-range kinodynamically constrained trajectory from the UAV's starting point to its goal. This problem is complicated by the fact that the space of possible control actions is extremely large, and that simplifications that reduce its dimensionality while preserving feasibility and near optimality are challenging.

Two well-known methods that have been applied to this problem are Probabilistic Road Maps [1] (PRMs) and Rapidly-exploring Random Trees [2] (RRTs). These methods reduce the dimensionality of the problem by sampling the possible control actions, but the resulting trajectories are generally not optimal. This work presents a new method for optimizing kinodynamically constrained trajectories that is near-optimal (shown to be within 3% on average for several examples) and computationally tractable.

Mixed-Integer Linear Programming (MILP) is the key element of this new approach. MILP extends linear programming to include variables that are constrained to take on integer or binary values. These variables can be used to add logical constraints into the optimization problem [3, 4], such as obstacle and collision avoidance [5, 6, 7]. Recent improvements in combinatorial optimization and computer speed make MILP a feasible tool for trajectory optimization. To reduce the computation time required by MILP for large trajectories, it can been used in a receding horizon framework, such as the basic approach presented in Ref. [5].

In general, receding horizon control (also called Model Predictive Control) designs an input trajectory that optimizes the plant's output over a period of time, called the *planning horizon*. The input trajectory is implemented over the shorter *execution horizon*, and the optimization is performed again starting from the state that is reached. This re-planning incorporates feedback to account for disturbances and plant modeling errors. In this problem setting, computation can be saved by applying MILP in a receding horizon framework to design a series of short trajectories to the goal instead of one long trajectory, since the computation required to solve a MILP problem grows non-linearly with its size.

One approach to ensuring that the successively planned short trajectories actually reach the goal is to minimize some estimate of the cost to go from the plan's end, or terminal point, to the goal. However, it is not obvious how to find an accurate cost-to-go estimate without planning the trajectory all the way to the goal, increasing the computation required. The approach suggested in [5] used an estimate of the time-to-go from endpoint of the plan to the goal that was not cognizant of obstacles in this interval. This terminal penalty led to the aircraft becoming trapped behind obstacles. Control Lyapunov Functions have been used successfully as terminal penalties in other problem settings [8], but these are also incompatible with the presence of obstacles in the environment. This paper investigates a cost-to-go

[1] The research was funded in part under DARPA contract # N6601-01-C-8075 and Air Force grant # F49620-01-1-0453.
[2] Research Assistant john_b@mit.edu
[3] Research Assistant, arthurr@mit.edu
[4] Associate Professor, jhow@mit.edu

function based on a modified version of Dijkstra's Algorithm applied to a visibility graph representation of the environment.

2 Fixed Horizon Minimum Time Controller

A minimum arrival time controller using MILP over a fixed planning horizon was presented in Ref. [7]. At time step k it designs a series of control inputs $\{\mathbf{u}(k+i) \in \mathbf{R}^2 : i = 0, 1, \ldots, L-1\}$, that give the trajectory $\{\mathbf{x}(k+i) \in \mathbf{R}^2 : i = 1, 2, \ldots, L\}$. Constraints are added to specify that one of the L trajectory points $\mathbf{x}(k+i) = [x_{k+i,1} \;\; x_{k+i,2}]^T$ must equal the goal $\mathbf{x}_{\text{goal}}$. The optimization minimizes the time along this trajectory at which the goal is reached, using L binary decision variables $\mathbf{b}_{\text{goal}} \in \{0,1\}$ as

$$\min_{\mathbf{u}(\cdot)} \; \phi_1(\mathbf{b}_{\text{goal}}, \mathbf{t}) = \sum_{i=1}^{L} b_{\text{goal,i}} t_{\text{i}} \tag{1}$$

subject to

$$\begin{aligned} x_{k+i,1} - x_{\text{goal},1} &\leq \;\; R(1 - b_{\text{goal,i}}) \\ x_{k+i,1} - x_{\text{goal},1} &\geq -R(1 - b_{\text{goal,i}}) \\ x_{k+i,2} - x_{\text{goal},2} &\leq \;\; R(1 - b_{\text{goal,i}}) \\ x_{k+i,2} - x_{\text{goal},2} &\geq -R(1 - b_{\text{goal,i}}) \end{aligned} \tag{2}$$

$$\sum_{i=1}^{L} b_{\text{goal,i}} = 1 \tag{3}$$

where R is a large positive number, and t_{i} is the time at which the trajectory point $\mathbf{x}(k+i)$ is reached. When the binary variable $b_{\text{goal,i}}$ is 0, it relaxes the arrival constraint in Eqn. 2. Eqn. 3 ensures that the arrival constraint is enforced once.

To include collision avoidance in the optimization, constraints are added to ensure that none of the L trajectory points penetrate any obstacles. Rectangular obstacles are used in this formulation, and are described by their lower left corner $(u_{\text{low}}, v_{\text{low}})$ and upper right corner $(u_{\text{high}}, v_{\text{high}})$. To avoid collisions, the following constraints must be satisfied by each trajectory point

$$\begin{aligned} x_{k+i,1} &\leq u_{\text{low}} \;\; + R\; b_{\text{in},1} \\ x_{k+i,1} &\geq u_{\text{high}} - R\; b_{\text{in},2} \\ x_{k+i,2} &\leq v_{\text{low}} \;\; + R\; b_{\text{in},3} \\ x_{k+i,2} &\geq v_{\text{high}} - R\; b_{\text{in},4} \end{aligned} \tag{4}$$

$$\sum_{\text{j}=1}^{4} b_{\text{in,j}} \leq 3 \tag{5}$$

The j^{th} constraint is relaxed if $b_{\text{in,j}} = 1$, and enforced if $b_{\text{in,j}} = 0$. Eqn. 5 ensures that at least one constraint in Eqn. 4 is active for the trajectory point. These constraints are applied to all trajectory points $\{\mathbf{x}(k+i) : i = 1, 2, \ldots, L\}$. Note that the obstacle avoidance constraints are not applied between the trajectory points for this discrete-time system, so small incursions into obstacles are possible. As a result, the obstacle regions in the optimization must be slightly larger that the real obstacles to allow for this margin.

The trajectory is also constrained by discretized dynamics, which model a fixed-wing aircraft with limited speed and turning rate. The latter is represented by a limit on the magnitude of the turning force $\mathbf{u}(k+i)$ that can be applied [7].

$$\begin{bmatrix} \dot{\mathbf{x}}(k+i+1) \\ \mathbf{x}(k+i+1) \end{bmatrix} = A \begin{bmatrix} \dot{\mathbf{x}}(k+i) \\ \mathbf{x}(k+i) \end{bmatrix} + B\mathbf{u}(k+i) \tag{6}$$

subject to the linear approximations of velocity and force magnitude limits

$$\ell_a(\dot{\mathbf{x}}(k+i)) \leq \dot{x}_{\max} \tag{7}$$

$$\ell_a(\mathbf{u}(k+i)) \leq u_{\max} \tag{8}$$

The constraints in Eqns. 7 and 8 make use of an approximation $\ell_a(\mathbf{r})$ of the length of a vector $\mathbf{r} = (x_1, x_2)$

$$\ell_a(\mathbf{r}) = s : s \geq L_1 x_1 + L_2 x_2, \; \forall \; (L_1, L_2) \in \mathcal{L} \tag{9}$$

where $\mathcal{L}$ is a finite set of unit vectors whose directions are distributed from $0^\circ - 360^\circ$.

This formulation finds the minimum arrival time trajectory. Experience has shown that the computational effort required to solve this optimization problem can grow quickly and unevenly with the product of the length of the trajectory to be planned and the number of obstacles to be avoided [5, 7]. However, as discussed in the following sections, a receding horizon approach can be used to design large-scale trajectories.

3 Simple Terminal Cost Formulation

In order to reduce the computational effort required, MILP has been applied within a receding horizon framework. To enable a more direct comparison of the effects of the terminal penalty, the following provides a brief outline of the receding horizon approach suggested in Ref. [5]. The MILP trajectory optimization is repeatedly applied over a moving time-window of length H_p. The result is a series of control inputs $\{\mathbf{u}(k+i) \in \mathbf{R}^2 : i = 0, 1, \ldots, H_p - 1\}$, that give the trajectory $\{\mathbf{x}(k+i) \in \mathbf{R}^2 : i = 1, 2, \ldots, H_p\}$. The first part of this input trajectory, of length $H_e < H_p$, is executed before a new trajectory is planned. The cost function of this optimization is the terminal penalty $\phi_2(\mathbf{x}(k+H_p))$, which finds the 1-norm of the distance between the trajectory's end point and the goal. The formulation is piecewise-linear and can be included in a MILP using slack variables as

$$\min_{\mathbf{u}(\cdot)} \; \phi_2(\mathbf{x}(k+H_p)) = \ell_b(\mathbf{x}_{\text{goal}} - \mathbf{x}(k+H_p)) \tag{10}$$

where $\ell_b(\mathbf{r})$ evaluates the sum of the absolute values of the components of $\mathbf{r}$. Obstacle avoidance and dynamics constraints are also added. This formulation

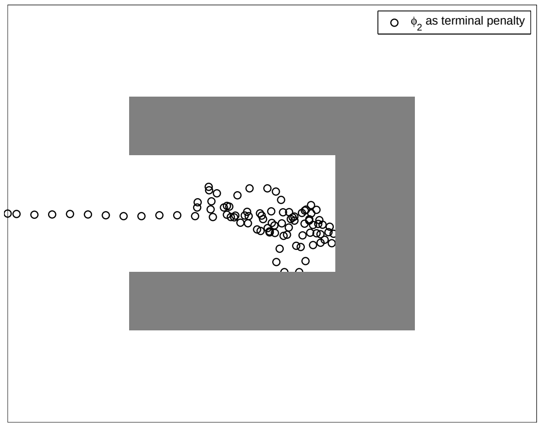


Fig. 1: Starting point at left, goal at right. Circles show trajectory points. Receding horizon controller using simple terminal penalty ϕ_2 and $H_p = 12$ becomes entrapped and fails to reach the goal.

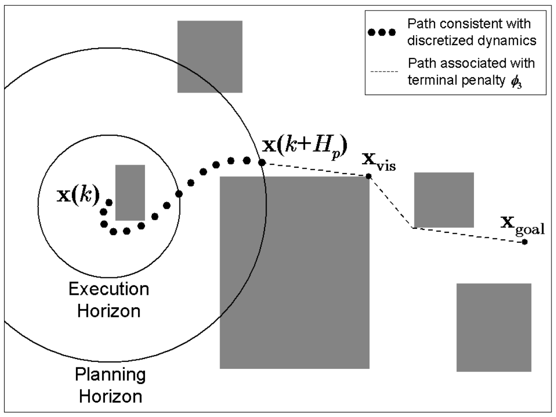
is equivalent to the fixed horizon controller when the horizon length is just long enough to reach the goal. However, when the horizon length does not reach the goal, the optimization minimizes the approximate distance between the trajectory's terminal point and the goal. This choice of terminal penalty can prevent the aircraft from reaching the goal when the approximation does not reflect the length of a *flyable path*. This occurs if the line connecting $\mathbf{x}(k + H_p)$ and the goal penetrates obstacles. This problem is especially apparent when the path encounters a concave obstacle, as shown in Fig. 1. If the terminal point that minimizes the distance to the goal is within the concavity behind an obstacle, then the controller will plan a trajectory into the concavity. If the path out of the concavity requires a temporary increase in the 1-norm distance to the goal, the aircraft can become trapped behind the obstacle. This is comparable to the entrapment in local minima that is possible using potential field methods.

4 Improved Receding Horizon Control Strategy

This section presents a novel method for approximating the time-to-go and shows how this can be implemented in a MILP program, using only linear and binary variables. This approach avoids the difficulties associated with nonlinear programming, such as choosing a suitable initial guess for the optimization.

4.1 Control Architecture

The control strategy is comprised of a cost estimation phase and a trajectory design phase. The cost estimation phase computes a compact "cost map" of the approximate minimum time-to-go from a limited set of points to the goal. The cost estimation phase is performed once for a given obstacle field and position of the goal, and would be repeated if the environment changes. The trajectory designer uses this cost map information in the terminal penalties of the receding horizon optimization to design a series of short trajectory segments that are followed until the goal is reached. This division of computation between the cost estimation and trajectory design phases enables the trajectory optimization to use only linear relationships. An example of a result that would be expected from the trajectory design phase is shown schematically in Fig. 2. In this phase, a trajectory consistent with discretized aircraft dynamics is designed from $\mathbf{x}(k)$ over a fine resolution planning horizon with length H_p steps. The trajectory is optimized using MILP to minimize the cost function assessed at the plan's terminal point $\mathbf{x}(k + H_p)$. This cost estimates the time to reach the goal from this point as the time to move from $\mathbf{x}(k + H_p)$ to a visible point $\mathbf{x}_{\text{vis}}$, whose cost-to-go was previously estimated, plus the cost-to-go estimate C_{vis} for $\mathbf{x}_{\text{vis}}$. As described in Section 4.2, C_{vis} is estimated using a coarser model of the aircraft dynamics that can be evaluated very quickly. Only the first H_e steps are executed before a new plan is formed starting from $\mathbf{x}(k + H_e)$.



Path consistent with discretized dynamics
Path associated with terminal penalty ϕ_3
$\mathbf{x}(k+H_p)$
$\mathbf{x}_{\text{vis}}$
$\mathbf{x}(k)$
$\mathbf{x}_{\text{goal}}$
Execution Horizon
Planning Horizon

Fig. 2: Resolution Levels of the Planning Algorithm

The use of two dynamics models with different levels of resolution exploits the trajectory planning problem's structure. On a long time-scale, a successful controller need only decide which combination of obstacle gaps to pass through in order to take the shortest dynamically feasible path. However, on a short time-scale, a successful controller must plan the dynamically feasible time-optimal route around the nearby obstacles to pass through the chosen gaps. The different resolution levels of the receding horizon controller described above allow it to make decisions on these two levels, without performing additional computation to "over plan" the trajectory to an unnecessary level of detail.

The cost estimation is performed in MATLAB. It produces a data file containing the cost map in the AMPL language, and an AMPL model file specifies the form of the cost function and constraints. The CPLEX optimization program is used to solve the MILP problem and outputs the resulting input and position trajectory. MATLAB is used to simulate the execution of this trajectory up to $\mathbf{x}(k+H_e)$, which leads to a new trajectory optimization problem with an updated starting point.

4.2 Computation of Cost Map

The shortest path around a set of polygonal obstacles to a goal, without regard for dynamics, is a series of joined line segments that connect the starting point, possibly obstacle vertices, and the goal. To find this path, a visibility graph can be formed whose nodes represent these points. Edges are added between pairs of nodes if the points they represent can be connected by a line that does not penetrate any obstacles. The visibility graph is searched using Dijkstra's Single Source Shortest Path Algorithm [9], starting at the goal, to find the shortest path from the each node of the graph to the goal, and the corresponding distances.

This work is concerned with minimum arrival time paths, not minimum distance paths. It is possible that the fastest path is not necessarily the shortest: longer paths can be faster if they require fewer sharp turns because the aircraft must slow down to turn sharply. This effect can be approximately accounted for by modifying Dijkstra's Algorithm. When Dijkstra's Algorithm examines an edge to see if it should be added to the tree of shortest paths being grown from the goal outwards, its length is divided by $\dot{x}_{\max}$, and a penalty is added that is proportional to the change in direction between the new edge and the next edge towards the goal in the tree. This approximately compensates for the reduction in aircraft speed necessary to make the turn between the two associated legs. With this modification, Dijkstra's Algorithm finds the approximate minimum time to reach the goal from each graph node.

In order to illustrate how the resulting cost map accounts for obstacles, their contribution to the cost is isolated in Fig. 3. To produce this graph, cost values were found over a fine grid of points in two fields of equal size, one with obstacles, and one without[1]. The two sets of cost values were subtracted to remove the contribution of straight line distance to costs in the obstacle field. Areas of larger difference are shown in Fig. 3 by darker shading. Note that the cost is increasing into the concave obstacle. This increase is crucial to avoiding the entrapment shown in Fig. 1.

[1]Cost values need not be found over such a fine grid to plan trajectories successfully. Since optimal large-scale trajectories tend to connect the starting point, obstacle vertices, and the goal, costs need only be found at these points. Many extra grid points are added here to more clearly demonstrate the trend in cost values.

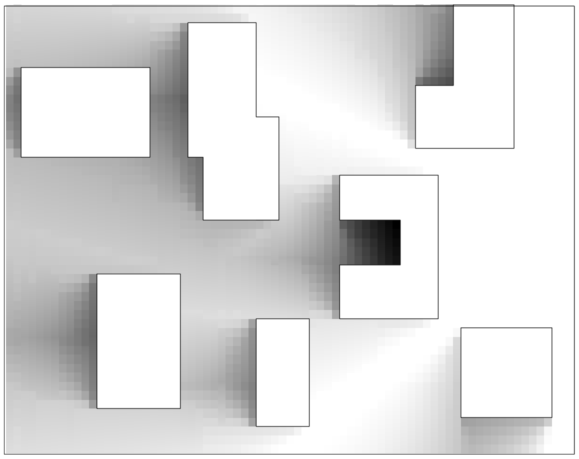


Fig. 3: Difference between actual cost at various points in an obstacle field and cost in same region with no obstacles, showing the effects of obstacles on cost values. Goal is at center right.

4.3 Modified MILP Problem

The results of the cost estimation phase are provided to the trajectory design phase as pairs of a position where the approximate cost-to-go is known and the cost at that point $(\mathbf{x}_{\text{cost,j}}, C_{\text{j}})$. This new formulation includes a significantly different terminal cost that is a function of $\mathbf{x}(k+H_p)$, and $(\mathbf{x}_{\text{vis}}, C_{\text{vis}})$, a pair from the cost estimation phase. The optimization seeks to minimize the time to fly from $\mathbf{x}(k+H_p)$ to the goal by choosing $\mathbf{x}(k+H_p)$ and the pair $(\mathbf{x}_{\text{vis}}, C_{\text{vis}})$ that minimize the time to fly from $\mathbf{x}(k+H_p)$ to $\mathbf{x}_{\text{vis}}$, plus the estimated time to fly from $\mathbf{x}_{\text{vis}}$ to $\mathbf{x}_{\text{goal}}$, C_{vis},

$$\min_{\mathbf{u}(\cdot)} \ \phi_3(\mathbf{x}(k+H_p), \mathbf{x}_{\text{vis}}, C_{\text{vis}}) = \frac{\ell_a(\mathbf{x}_{\text{vis}} - \mathbf{x}(k+H_p))}{v_{\max}} + C_{\text{vis}} \tag{11}$$

A key element in the algorithm is that the optimization is not free to choose $\mathbf{x}(k+H_p)$ and $\mathbf{x}_{\text{vis}}$ independently. Instead, $\mathbf{x}_{\text{vis}}$ is constrained to be visible from $\mathbf{x}(k+H_p)$. Note that visibility constraints are, in general, nonlinear because they involve checking whether every point along a line is outside of all obstacles. Because these nonlinear constraints cannot be included in a MILP problem, they are approximated by constraining a discrete set of interpolating points between $\mathbf{x}(k+H_p)$ and $\mathbf{x}_{\text{vis}}$ to lie outside of all obstacles. These interpolating points are a portion τ of the distance along the line-of-sight between $\mathbf{x}(k+H_p)$ and $\mathbf{x}_{\text{vis}}$

$$\forall\, \tau \in \mathcal{T} : [\mathbf{x}(k+H_p) + \tau \cdot (\mathbf{x}_{\text{vis}} - \mathbf{x}(k+H_p))] \notin \mathcal{X}_{\text{obst}} \tag{12}$$

where $\mathcal{T} \subset [0,1]$ is a discrete set of interpolation distances and $\mathcal{X}_{\text{obst}}$ is the obstacle space.

The visibility constraint ensures that the length of the line between $\mathbf{x}(k+H_p)$ and $\mathbf{x}_{\text{vis}}$ is a good estimate of the length of a flyable path between them, and ultimately

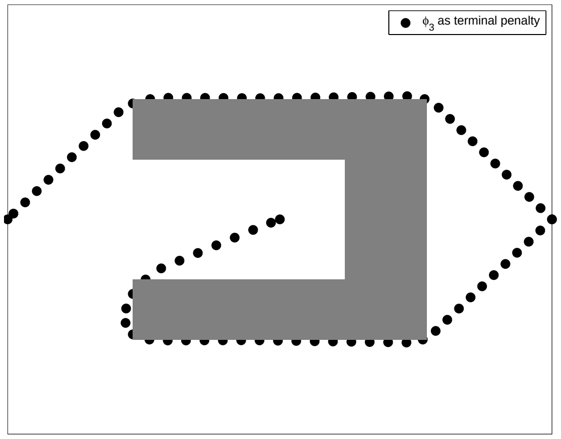


Fig. 4: Trajectories designed using receding horizon controller with ϕ_3 terminal penalty avoid entrapment. Trajectories start at left and at center, goal is at right. Circles show trajectory points. $H_p = 12$.

that the terminal penalty is a good estimate of the time to reach the goal from $\mathbf{x}(k + H_p)$. The interpolating points are constrained to lie outside obstacles in the same way that the trajectory points are constrained to lie outside obstacles in the previous formulations (see Eqns. 4 and 5), so it is possible that portions of the line-of-sight between interpolating points penetrate obstacles. However, if a sufficient number of interpolating points is used, the line-of-sight will only be able to "cut corners" of the obstacles. In this case, the extra time required to fly around the corner is small, and the accuracy of the terminal penalty is not seriously affected.

The values of the position $\mathbf{x}_{\text{vis}}$ and cost C_{vis} are evaluated using the binary variables $\mathbf{b}_{\text{cost}} \in \{0, 1\}$ and the n points on the cost map as

$$\mathbf{x}_{\text{vis}} = \sum_{j=1}^{n} b_{\text{cost,j}} \mathbf{x}_{\text{cost,j}} \qquad (13)$$

$$C_{\text{vis}} = \sum_{j=1}^{n} b_{\text{cost,j}} C_{\text{j}} \qquad (14)$$

$$\sum_{j=1}^{n} b_{\text{cost,j}} = 1 \qquad (15)$$

Obstacle avoidance constraints (Eqns. 4 and 5) are enforced without modification at $\{\mathbf{x}(k + i) : i = 1, 2, \ldots, H_p\}$. The dynamics model (Eqn. 6), the velocity limit (Eqn. 7), and the control force limit (Eqn. 8) are also enforced in this formulation. This provides a completely linear receding horizon formulation of the trajectory design problem.

5 Results

The following examples demonstrate that the new receding horizon control strategy provides trajectories that are close to time-optimal and avoid entrapment, while maintaining computational tractability.

5.1 Avoidance of Entrapment

In order to test the performance of the improved cost penalty around concave obstacles, the improved terminal penalty ϕ_3 was applied to the obstacle field considered in Section 3, and the resulting trajectories are shown in Fig. 4. The new cost function captures the difference between the distance to the goal and the length of a feasible path to the goal, which allows the receding horizon controller to plan trajectories that consistently reach the goal.

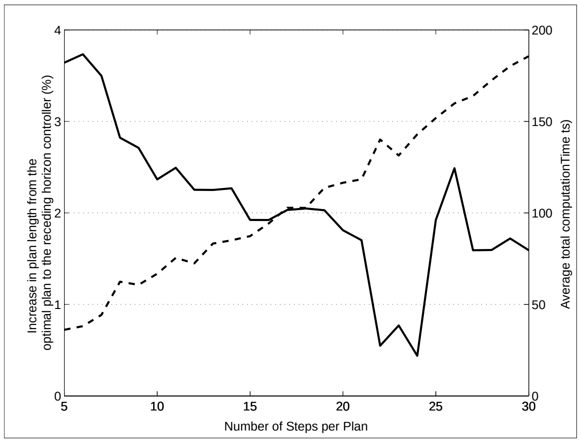


Fig. 5: The Effects of Plan Length. Increase in arrival time from optimal to that found by receding horizon controller is plotted with a solid line. Average total computation time is plotted with a dashed line.

5.2 Optimality

The computational effort required by the receding horizon control strategy is significantly less than that of the fixed horizon controller because its planning horizon is much shorter. However, for the same reason, global optimality is not guaranteed. To examine this tradeoff, a set of random obstacle fields was created, and a trajectory to the goal was planned using both the receding and fixed horizon controllers. The receding horizon controller was applied several times to each problem, each time with a longer planning horizon. The results are shown in Fig. 5. The extra number of time steps in the receding horizon controller's trajectory is plotted as a percentage of the minimum number of steps found using the fixed horizon controller, averaged over several obstacle fields. The plot shows that, on average, the receding horizon controller is within 3% of the optimum for a planning horizon longer than 7 time steps. The average total computation time for the receding horizon controller is also plotted, showing that the increase in computation time is roughly linear with plan length.

5.3 Computation Savings

The effects of problem complexity on computation time were also examined by timing the fixed and receding horizon controllers' computation on a 1 GHz PIII computer. The complexity of a MILP problem is related to its number of binary variables, which grows with the product of the number of obstacles and number of steps to be planned. A series of obstacle fields was

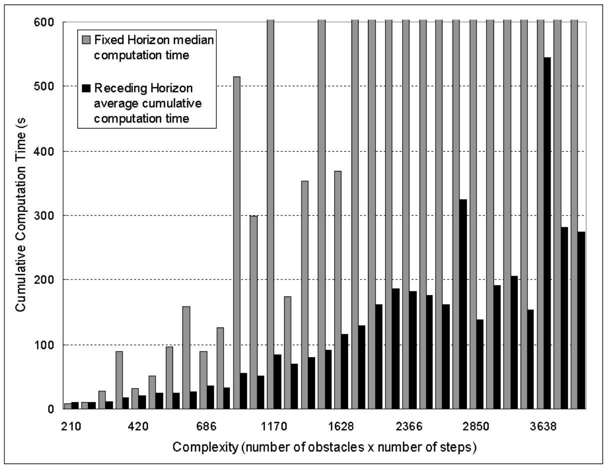


Fig. 6: Cumulative Computation Time vs. Complexity

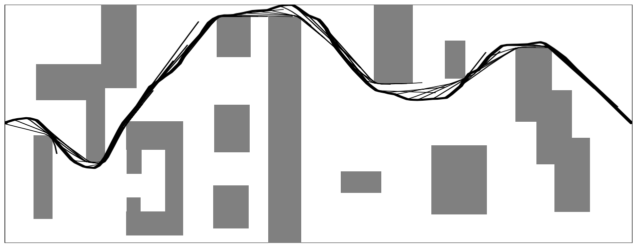


Fig. 7: Sample Long Trajectory Designed using Receding Horizon Controller. Executed trajectory (plan plus velocity disturbance) shown with thick line, planned trajectory segments shown with thin lines.

created with several different values of this complexity metric, and a trajectory to the goal was planned using both the receding and fixed horizon controllers. The optimization of each plan was aborted if the optimum was not found in 600 seconds. The results are shown in Fig. 6, which gives the average cumulative computation time for the receding horizon controller, and the median computation time for the fixed horizon controller. The median computation time for the fixed horizon controller was over 600 seconds for all complexity levels over 1628. At several complexity levels for which its median computation time was below 600 seconds, the fixed horizon controller also failed to complete plans. The cumulative time required by the receding horizon controller to design all the trajectory segments to the goal was less than this time limit for every problem. All but the first of its plans can be computed during execution of the previous, so the aircraft can begin moving towards the goal much sooner.

Next, an extremely large problem, with a complexity of 6636, was attempted with the receding horizon controller. It successfully designed a trajectory that reached the goal in 316 time steps, in a cumulative computation time of 313.2 seconds. This controller took 2.97 seconds on average to design one trajectory segment. The fixed horizon controller could not solve this problem in 1200 seconds of computation time. A trajectory through the same obstacle field was also planned in the presence of velocity disturbances, causing the followed trajectory to differ significantly from each of the planned trajectory segments. By designing each trajectory segment from the state that is actually reached, the receding horizon controller compensates for the disturbance. The executed trajectory and planned trajectory segments are shown in Fig. 7.

6 Conclusions

This paper presents a new algorithm for designing long-range kinodynamically constrained trajectories for fixed-wing UAVs. It is based on MILP optimization within a receding horizon control framework. A novel terminal penalty for the receding horizon optimization is computed by finding a cost map for the environment, and connecting the aircraft trajectory over the planning horizon to the cost map. The resulting MILP problem can be solved with commercially available optimization software. Simulation results show that: (a) the receding horizon controller plans trajectories whose arrival times are within 3% of optimal; (b) the controller can successfully solve complex trajectory planning problems in practical computation times; and (c) the controller avoids entrapment behind obstacles.

References

[1] Kavraki, L. E., Lamiraux, F., and Holleman, C., "Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces," *IEEE Transactions on Robotics and Automation*, 12(4):566-580.

[2] LaValle, S. M. and Kuffner, J. J., "Randomized Kinodynamic Planning," *IEEE International Conference on Robotics and Automation*, July, 1999.

[3] Floudas, C. A., "Nonlinear and Mixed-Integer Programming – Fundamentals and Applications," Oxford University Press, 1995.

[4] Williams, H. P., and Brailsford, S. C., "Computational Logic and Integer Programming," in *Advances in Linear and Integer Programming,* Editor J. E. Beasley, Clarendon Press, Oxford, 1996, pp. 249–281.

[5] Schouwenaars, T., De Moor, B., Feron, E. and How, J., "Mixed Integer Programming for Multi-Vehicle Path Planning," *ECC*, 2001.

[6] Richards, A., How, J., Schouwenaars, T., Feron, E., "Plume Avoidance Maneuver Planning Using Mixed Integer Linear Programming" AIAA *GN&C Conf.*, 2001.

[7] Richards, A. and How, J., "Aircraft Trajectory Planning With Collision Avoidance Using Mixed Integer Linear Programming", to appear at *ACC*, 2002.

[8] Jadbabaie, A., Primbs, J. and Hauser, J. "Unconstrained receding horizon control with no terminal cost", presented at the *ACC*, Arlington VA, June 2001.

[9] Cormen, T. H., Leiserson, C. E., Rivest, R. L.. "Introduction to Algorithms," MIT Press, 1990.