Online Trajectory Planning for UAVs Using Mixed Integer Linear Programming

by

Kieran Forbes Culligan

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

August 2006

Author .....
Department of Aeronautics and Astronautics
August 11, 2006

Certified by .....
Jonathan P. How
Associate Professor
Thesis Supervisor

Accepted by .....
Jaime Peraire
Chairman, Department Committee on Graduate Students

Online Trajectory Planning for UAVs Using Mixed Integer Linear Programming

by

Kieran Forbes Culligan

Submitted to the Department of Aeronautics and Astronautics
on August 11, 2006, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

This thesis presents a improved path planner using mixed-integer linear programming (MILP) to solve a receding horizon optimization problem for unmanned aerial vehicles (UAV's). Using MILP, hard constraints for obstacle and multi-vehicle avoidance as well as an approximation of vehicle dynamics are included into the formulation. The complete three dimensional formulation is described. The existing MILP framework has been modified to increase functionality, while also attempting to decrease solution time.

A variable time step size, linear interpolation points, and horizon minimization techniques are used to enhance the capability of the online path planner. In this thesis, the concept of variable time steps is extended to the receding horizon, non-iterative MILP formulation. Variable time step sizing allows the simulation horizon time to be lengthened without increasing solve time too dramatically. Linear interpolation points are used to prevent solution trajectories from becoming overly conservative. Horizon minimization decreases solve time by removing unnecessary obstacle constraints from the the problem.

Computer simulations and test flights on an indoor quadrotor testbed shows that MILP can be used reliably as an online path planner, using a variety of different solution rates. Using the MILP path planner to create a plan ten seconds into the future, the quadrotor can navigate through an obstacle-rich field with MILP solve times under one second. Simple plans in obstacle-spare environments are solved in less than 50ms. A multi-vehicle test is also used to demostrate non-communicating deconfliction trajectory planning using MILP.

Thesis Supervisor: Jonathan P. How
Title: Associate Professor

Acknowledgments

This research was completed with the help of many individuals. First I would like to express the most sincere thanks to my advisor Prof. Jonathan How. He was brave enough to take me on as a green undergraduate, and it was with his careful supervision and insightful comments that I found the capability to finish this thesis. The faculty at MIT as a whole has pushed me in my personal and academic pursuits in a way I never imagined possible. The constant reach for intellectual excellence is unparalleled at another other institution, and I will surely miss it.

Department administrators Barbara Lechner and Marie Stuppard, the miracle makers, kept me on track with all of the MIT logistics that I could never be able to handle on my own. Without their assistance I would never have made it this far through the Institute. Administrative assistant Kathryn Fischer was just as helpful, making me feel welcome as a part of Prof. How's research group and setting up meetings whenever I needed them.

I am also extremely grateful to Dr. Robert Miller, manager of GNC and Air Vehicle Autonomy at Northrop Grumman Unmanned Systems. He introduced me to the MILP path planning formulation during a summer internship. Cedric Ma, also at Northrop Grumman, offered his MILP expertise and served as a valuable resource. Thanks to Dave Grosher and Dan Marthaler for offering some fun lunchtime commentary during my visits to San Diego.

My time at MIT was filled with some great times: various adventures with Chris Mitchell, school trips to Long Beach with Ben Stewart, presenting sports research in Munich with David Walfisch, a quick cycling adventure through Europe with my high school buddy Dan Cronin, being part of Unified's "back row slackers," and, of course, being part of the cycling team. The cycling team has definitely kept me sane during my years at MIT. A few hours on the bike always seemed to cheer me up a bit. Training through the winters was miserably fun, and the race weekends were simply epic. No other racing team will ever be as much fun.

The push to finish this thesis was no small effort, and it was the presence of some

great colleagues that made it all worthwhile. Mario Valenti had an unfailing ability to help me out whenever I needed it. Luckily, he was never attacked by any MILP-controlled quadrotors, but there were some close calls. I am sure that Brett Bethke is able to get actual work done now that I am no longer around, constantly asking him Java and Linux-related question to no end. Yoshi Kuwata graciously spent time with me in my quest to improve the MILP formulation that he has helped develop. I am fortunate to have been surrounded by so many helpful and friendly people at the Aerospace Controls Lab, including: Luca Bertuccelli, Mehdi Alighanbari, Gaston Fiore, Dan Dale, Mike Matczynski, Pål Forus, Chun Sang Teo, Byunghoon Kim, Han-Lim Choi, and all of the space folks, too.

Finally, I would like to thank my family. Natasha has offered me more support than I ever could have asked for during such a challenging period. Sorry for using our summer travel time to finish this thesis! My Mom, Dad, and two brothers are the best family that a person could have. Thanks!

The quadrotor indoor test platform has been generously supported by the Boeing Company under the guidance of Dr. John Vian at the Boeing Phantom Works, Seattle. Research also funded in part by AFOSR grant FA9550-04-1-0458.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Unmanned Aerial Vehicles

Unmanned aerial vehicles (UAV's) were first developed in the wake of World War I. These aircraft were designed to act as "aerial torpedoes," now referred to as cruise missiles [57, 33]. By World War II, UAV's were flown as targets for practicing anti-aircraft gunners. For several decades after the war, UAV's continued to be used as target drones and for a few reconnaissance missions [31]. But even through the 1970's, many UAV's were little more than normal aircraft outfitted with a radio controller and servos on the control mechanisms. The introduction of two successful products, General Atomic's RQ-1 Predator and Northrop Grumman's RQ-4 Global Hawk, spawned a wave of development in the field beginning in the 1990's. Today, development and deployment of UAV's is growing at an unprecedented rate. Both military and civilian customers are seeking UAV's that have capabilities that could not have even been imagined just a decade ago. Multi-day missions, aggressive maneuvering, and autonomous ship landings are becoming a reality in the UAV world. The overarching desire is to make new UAV's as autonomous as possible [19].

Autonomy is not a single feature, but a holistic measure of the vehicle's ability to perform tasks and make decisions without human interaction [55]. Types of autonomy include sensor fusion, communication, path planning, and task allocation. This thesis will focus on the path planning element of autonomy. That is, how to determine an

optimal path for the air vehicle while meeting certain objectives and adhering to a set of constraints.

1.2 Trajectory Planning for Autonomous Aerial Vehicles

The need for online trajectory planning is strong today, but it has not always been that way. Since early UAV's were little more than advanced remote control aircraft, ground operators handled all aspects of the vehicle's flight. Even today, most unmanned vehicles in operation are either controlled remotely by pilots on the ground or pre-programmed with a set of waypoints to follow. This approach to the control of unmanned aircraft will become infeasible as the number of unmanned aircraft grows and as the missions they are tasked with become more challenging and dynamic. The path planning component of vehicle autonomy involves much more than simply commanding a vehicle to fly from point A to point B. Online trajectory planning allows for the aircraft to actively modify its flight path during the course of a mission, without relying on the intervention of a human operator. Such a path planner must have the ability to generate trajectories that will enable the vehicle to avoid obstacles, complete tasks, minimize fuel burn, and meet other objectives, all while taking into account dynamic constraints.

The path planner is subject to many constraints and objectives. The path planner has to account for environmental features such as ground terrain, buildings, radar sites, even other aircraft. It also considers aspects related to the overall mission such as a list of mission waypoints, or a limit on altitude. The vehicle dynamics play an important part in the problem since they define the feasibility of a particular trajectory. Autonomous trajectory planning must produce a safe, efficient flight path for the vehicle. In most scenarios, the path cannot be calculated in advance. The path planner must run in real-time on the vehicle, enabling the vehicle to respond to dynamic events.

Figure 1-1: Northrop Grumman's MQ-8B Fire Scout

1.3 Literature Overview

Trajectory planning has been a well-researched area in the field of robotics and artificial intelligence. The aerospace industry has also been interested in the path planning problem for decades, with the initial intention of developing pre-calculated trajectories for missiles, spacecraft, and supersonic aircraft. Early research focused on using optimal control and nonlinear programming to develop offline plans. A survey of methods is covered in a paper by Betts [10].

A brief review of salient topics regarding modern planning techniques is broken up into two sections. The first section discusses early contributions to the field and the second focuses on more modern developments using *model predictive control* (MPC).

1.3.1 Non-MPC Techniques

Leonard Euler's famous "Königsberg bridge" problem, posed in 1736, is one of the first investigations into modern path planning [11]. His work served as the foundation for what is now known a graph theory. Graph theory is used in many path planning alogrithms, most notably the classic "shortest path" algorithm. The underlying principle is that a set of all possible solutions is created, and the set is searched until the optimal (or near-optimal) solution is attained. Graph theory and graph search is still a popular topic in the computer science and artifical intelligence fields [13].

The "vehicle routing problem" tries to find the optimal path for a set of vehicles that must accomplish a list of tasks. Although intended for multi-vehicle problems, some solutions to the vehicle routing problems apply to the single vehicle path plan-

ning case. A survey of algorithms, is covered by Toth and Vigo [51], and Hwang [20]. One subset of the vehicle routing problem is the traveling salesman problem [35, 23], which is used for the foundation of many UAV task allocation algorithms [5].

The principle of *dynamic programming* was developed in 1953 by Bellman [6]. In this approach, the state space is discretized, and so the motion planning problem is reduced to connecting a finite set of states, each separated by a pre-defined Δt, in a dynamically feasible way. Dynamic programming has been used to develop obstacle avoidance trajectories for aircraft [29] and manage air traffic spacing [2]. Dynamic programming works for small problems but the search space and corresponding traversal time skyrockets as the problem size increases, an effect that Bellman himself reffered to as the "curse of dimensionality" [7].

1.3.2 Path Planning Using MPC

The principle of MPC, also referred to as *receding horizon control* (RHC), is to iteratively solve a finite time optimal control problem. Using an initial state, a trajectory is designed by minimizing some performance criterion over a fixed horizon, often subject to constraints on the inputs and outputs of the problem. By only having to optimize a solution from $t = t_0 \rightarrow t = (t_0 + \Delta T)$, the problem complexity is greatly reduced compared to an infinite time optimization. Survey papers by Bemporad and Morari [8] and Qin and Badgwell[34] cover the general principles related to MPC. MPC has been used to manage large industrial processes since the 1980's. For example, chemical plants use MPC in real-time to manage production rate and detect problems [17]. These problems can include hundreds and even thousands of states, requiring many minutes or hours to solve.

In an attempt to further simplify and speed up the MPC problem, an alternative option formulation has been developed. By posing the MPC problem as a *mixed-integer linear programming* (MILP) problem, it is possible to include hard constraints and logical statements in the optimization formulation. The movement from a non-linear to a linear framework makes the problem much less difficult from a computation standpoint. Air vehicle trajectory planning using MILP was first in-

troduced by Schoewenaars in [43, 45], and has been extended by many others. MILP has also been looked at as a feedback solution to the more classical control problem [39].

1.4 Statement of Contributions

This thesis introduces several important features that improve the overall usefulleness of an online MILP path planner. The overarching principle applied throughout is the idea of *variable MILP*. Many previous MILP implementations use problem statements that are largely the same from one timestep to the next. With *variable MILP*, the constraints and variables in place at each timestep are tailored in order to minimize complexity and improve robustness. Some papers have formulations that apply this principle in certain cases, but in this thesis it is used wherever possible. The real-time applicability of the MILP path planner is critical, so features that increase solve time by more than a few seconds are not considered in this thesis.

A complete three dimensional MILP formulation is provided everywhere applicable. In some instances, the changes required to go from a two dimension to a three dimension formulation are not immediately clear. Horizon minimization is among one of the techniques introduced that makes the MILP easier to solve by removing unnecessary constraints from the problem. The variable timestep approach, first introduced as part of the *iterative MILP* formulation [14], is used here in a receding horizon, non-iterative context. Variable time steps allow for a longer planning distance without adding decision variables, thereby lengthening the simulation time while not increasing complexity. Linear interpolation points are used to keep obstacles from being over-expanded, a shortcoming in earlier MILP papers. A unique formulation for moving obstacles enables more intelligent path planning around ground and air vehicles, given only their current state information.

Critical changes have also been made to the implementation strategy for the MILP problem. Using a "warm-start" approach, execution time can be dramatically reduced at each solve iteration, especially for small problems that are executed at a high rate.

The MILP problem does not need to be built from the ground up every time. Instead, the existing formulation is modified and resolved.

Using a quadrotor testbed, the strategies mentioned above were test and modified to assure proper performance in an actual flight environment. This experimental component of the research was infinitely helpful in identifying real-word problems that are not apparent in computer simulations. Adding the techniques introduced in this thesis to the core MILP formulation results in a path planner with increased robustness and capabilities. At the same time, solve time is kept at a minimum, allowing for the path planner to run real-time on an air vehicle platform.

1.5 Thesis Outline

The beginning chapters of this thesis describe the mechanics of what is possible with MILP path planning. Chapter 2 introduces the use of MILP for trajectory planning, and applies it to the receding horizon path planning problem. The core constraints and formulations are introduced and some important complexity issues are addressed. Chapter 3 goes on the cover a more expanded set of features that can be used in a MILP path planner. Some new topics covered in this chapter include using linear interpolation points, and using more complex plant representations. Chapter 4 focuses on using MILP for nap-of-the-earth (NOE) flight and provides a new algorithm for sizing the resolution of the ground terrain according to the vehicle dynamics. Simulation results demonstrate the capability of using MILP to navigate through mountainous terrain. The later part of the thesis addresses issues related to implementation of the techniques described in the previous chapters. Chapter 5 discusses some of the software-specific elements regarding an implementation of the MILP path planning problem for both simulation and actual test flights on an indoor quadrotor test platform. A set of flight test are described, analyzed, and compared to simulated results in Chapter 6. Chapter 7 serves as a conclusion to the thesis, summarizing the important findings and offering some suggestions for future research.

Chapter 2

MILP Path Planning with Planning Horizon Minimization

2.1 Introduction

This chapter describes the basics mechanics of a MILP problem and goes on to explain the core MILP formulation used throughout this thesis. Some of MILP components described in this chapter are the same as used in previous implementations [44, 25, 41], but other features, such as the variable timesteps and horizon minimization, are modifications of existing techniques, and can significantly improve problem performance. More advanced and specific features are covered in Chapters 3 and 4.

2.2 Mixed Integer Linear Programming

Several methods have been developed to create trajectories for air vehicles that take into account vehicle dynamics, environment constraints, and mission objectives. Unfortunately, most of these methods cannot be solved fast enough to be used in real time. The computational complexity requires that solutions be calculated offline, but UAV's require a path planner capable of being run real-time onboard the vehicle. A MILP path planner uses the advantages of MPC (reduced problem size over a fixed horizon), and further simplifies the problem so that a solution can be solved in a

reasonable amount of time. By simplifying the problems so that it can be stated as a set of linear and binary constraints, the problem becomes much more tractable, and puts the problem statement into a form solvable by many robust mathematical programming solvers. Increased computer processor speeds, improved solver algorithms, and slimmer problem formulations allow for MILP problems to be solved onboard air vehicles, aiding in real-time navigation.

The demand for fast, robust MILP solvers is shared by many industries beyond aerospace. Finance and operations fields use MILP to solve a variety of complex problems. Commercial and open-source efforts are continually refining their MILP solvers, so this paper does not spend time trying to improve the solution algorithms. Instead, the focus is on the problem formulation. The mathematical programming solver CPLEX[21] is used throughout this thesis.

The goal of this thesis is to create a MILP that will solve quickly and produce a safe, but not overly conservative, trajectory towards a goal. Problem complexity is looked at holistically (solve time, number of decision variable, number of binaries, number of constraints, etc.), as an analytical perception of complexity for a MILP is not well defined. Linear programming is solvable in polynomial time, but the addition of binaries into the problem framework makes solution time less predictable [54]. While linear programming is a problem of class $\mathcal{NP}$, mixed integer linear programming is of class $\mathcal{NP}$-hard [18].

Increases in computer processing speeds and solution algorithms has allowed for addition of binary variables to the linear programming model [56]. This allows for logic and constraint activation within the MILP enabling non-convex constraints and logical constraints to be included in the problem.

A MILP has the following standard form [15]:

$$\text{minimize} \quad f(x) = c^T x \tag{2.1}$$

$$\text{subject to:} \quad b_L \leq Ax \leq b_U \tag{2.2}$$

$$x_L \leq x \leq x_U \tag{2.3}$$

where $A \in \mathbb{R}^{m \times n}$; $c, x, x_L, x_U \in \mathbb{R}^n$; $b_L, b_U \in \mathbb{R}^m$. A subset variables $x_I \in x$ are restricted to integer values, that is $x_I \in \mathbb{Z}^{n_I}$. This set of integer variables also includes the binary variables x_B, where $x_B \in \{0, 1\}$.

The initial two-dimensional MILP path planning formulation has since been extended in three dimensions [26] and new features have been added, such as the capability to solve minimum time problems [37] and minimize detectability by radar sites [12]. The MILP path planning formulation is supplemented with safety circle and return trajectories in [44, 47]. As features and capabilties are added to the core forumlation one issue becomes apparent: increased solve time. Some complex problems require minutes to solve, but are creating waypoints that are only suppossed to be a few seconds apart. An iterative solution method to combat this problem is proposed in [14]. MILP path planners have been implemented on vehicles in a few instances for both 2D [53, 40] and 3D [42, 48] path planning, but many of the MILP path planning papers only include simulation results and never test to see how an actual vehicle would follow the solution.

2.2.1 Receding Horizon

The trajectory planning problem is solved with a receding horizon for several reasons. First, planning out an entire mission is computationally intensive [36]. Second, there is no value in solving for the entire mission if the environment or mission plan is expected to change during the course of the mission. The receding hosizon implementation allows for constant revision and updates based on current information.

2.2.2 Time Discretization

Time is discretized in order to transform the trajectory optimization problem in to a MILP. In this thesis we will always use time $t_0 = 0$ as the initial time and $t_f = T_s$ as the total *simulation time.* The time t is divided into N pieces, where $\Delta t_n = t_n - t_{n-1}$ for $n = 1 \ldots N$. Thus, the total simulation time is:

$$T_s = \sum_{n=1}^{N} \Delta t_n \tag{2.4}$$

The majority of previous MILP implementations have used a fixed timestep $\Delta t_n = \Delta t$ between each timestep[44, 25], as illustrated in Figure 2-1. For this fixed timestep case, the total simulation time is simply: $T_s = \Delta t\, N$.

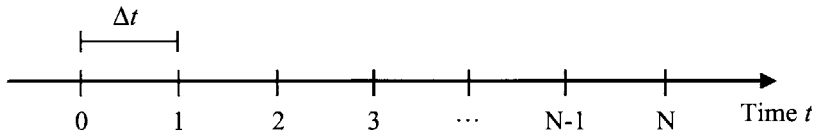


Figure 2-1: Fixed timesteps

Now let us assume that the Δt's are not fixed, such that Δt_n does not have to equal Δt_{n+1}. This idea of using a variable time step size is introduced in [14], as part of the *iterative MILP* formulation. A MILP problem was solved iteratively, creating more time steps with each iteration. Here will we used a pre-determined set of time step sizes, and solve the problem in one iteration. Using this method, it is possible to have the closer waypoint spaced tightly, and the more distant waypoints further apart (see Figure 2-2). This keeps the short-term plan from being too conservative, and develops a coarser plan for the distant waypoints, where we typically know less about the environment. Total simulation time T_s is increased without having to enlarge the first few timesteps or add extra timesteps to the problembgg(which means more decision variables and binaries). This is advantegous, because it means that the end of the plan will be closer to the goal state, decreasing the chance that the problem will get trapped by local minima. The simulations pictured in Figures 2-1 and 2-2 demonstrate this point. In Figure 2-1 the fixed timestep method is used. There are six timesteps each sized at 1 second ($\Delta t = [1, 1, 1, 1, 1, 1]$). The average solution time was 63ms. You can see that the vehicle gets stuck in a corner because the horizon time was too short. In Figure 2-2 a variable time step approach with linear interpolation points is used (linear interpolation points are covered later in Section 3.2). The timesteps were setup as $\Delta t = [1, 1, 2, 2, 6, 6]$. The average solution time

was 342ms. Using variable timesteps, the vehicle was able to extend the horizon time for $T_s = 6$ seconds to $T_s = 18$ seconds, and the result is that the vehicle avoided the dead-end corner and is able to eventually reach the goal state successfully. The difference between the two solution times may seem signficant, but take note that solving the same problem using 18 timesteps all spaced at 1 second had an average solution time of 1.67 seconds.

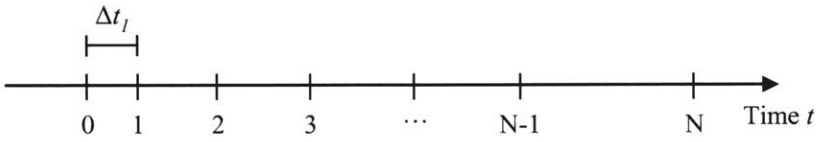


Figure 2-2: Variable time steps

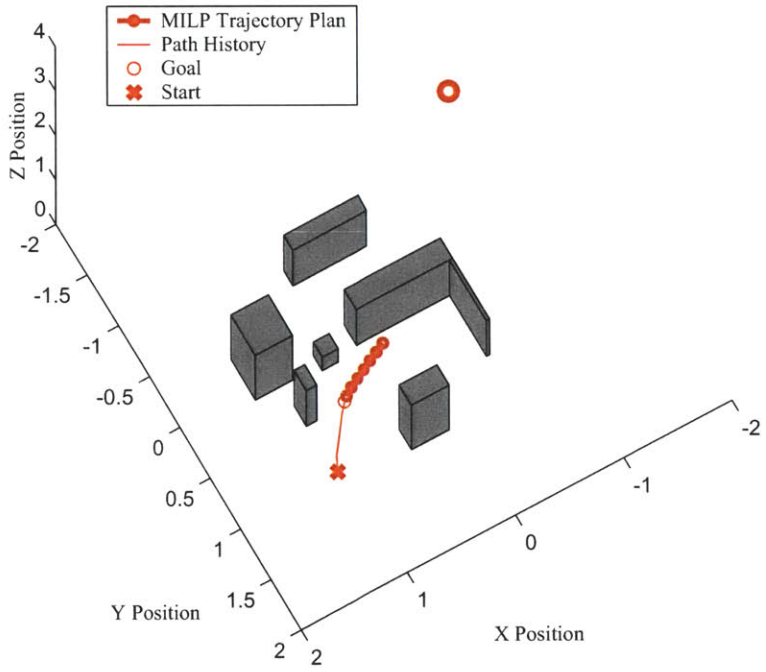


Figure 2-3: Simulation using fixed time steps of 1 second each. The seventh solution iteration is shown.

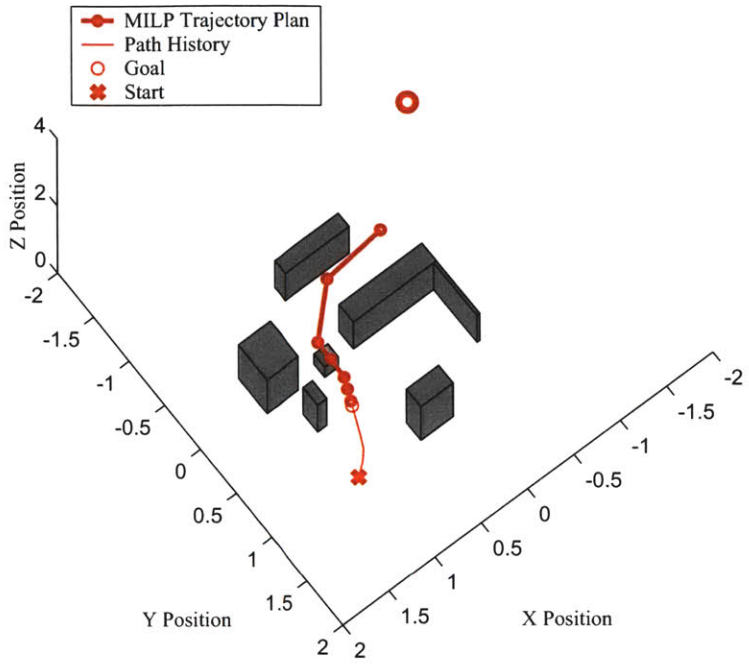


Figure 2-4: Simulation using variables time step sizing. The seventh solution iteration is shown.

2.2.3 Initial Conditions

The initial state condition $\mathbf{x}_0$ and $\mathbf{v}_0$ used in the MILP formulation is a predicted state based in the most recent known state information. Ideally, the initial condition should be the state of the vehicle when the MILP is *solved*. Because the vehicle will move while the MILP is being update and solved, it is necessary to predict the vehicle state just before the MILP is solved (see Figure 2-5). Based on the general complexity of the MILP path planner, you can experimentally determine an upperbound on the update and solve time for the MILP, which we will define as T_{gap}. By linearly interpolating the last known state information ($\mathbf{x}_{\text{last}}$ and $\mathbf{v}_{\text{last}}$) for time T_{gap}, the initial conditions for the MILP are:

$$\mathbf{x}_0 = \mathbf{x}_{\text{last}} + \mathbf{v}_{\text{last}} T_{\text{gap}} \tag{2.5}$$

$$\mathbf{v}_0 = \mathbf{v}_{\text{last}} \tag{2.6}$$

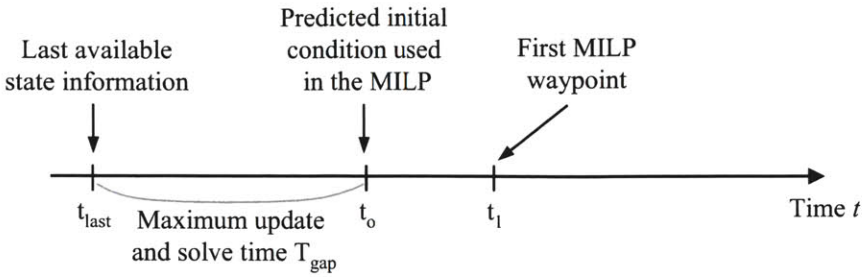


Figure 2-5: The initial condition used in the MILP problem is estimated to account for movement while the problem is being solved.

2.3 Vehicle Dynamics

The governing dynamics for any air vehicle are typically non-linear, and so the dynamics must be approximated as being either linear or piece-wise linear in order to include them in the MILP. This section describes the system of equation used to represent vehicle dynamics, including bounds on acceleration, velocity, climb/descend rate, and climb/descend acceleration rate.

2.3.1 State Equations

The state quations used in the MILP are a linearized, discretized version of a vehicle plant model. The general form for a linear time-invariant (LTI) system with a continuous-time plant is:

$$\dot{\mathbf{s}} = A_c\mathbf{s} + B_c\mathbf{u} \tag{2.7}$$

For the majority of this thesis we will assume double integrator plant dynamics subject to a maximum speed and maximum climb/descent rate. In the case of the double integrator, the dynamics are written as:

$$\frac{d}{dt}\begin{bmatrix}\mathbf{x}\\ \mathbf{v}\end{bmatrix} = \begin{bmatrix}0 & I_3\\ 0 & 0\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ \mathbf{v}\end{bmatrix} + \begin{bmatrix}0\\ \mathbf{u}\end{bmatrix} \tag{2.8}$$

where $\mathbf{x}$ is the position vector defined as $\mathbf{x} = \begin{bmatrix} x & y & z \end{bmatrix}^T$, $\mathbf{v}$ is the velocity vector $\mathbf{v} = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix}^T$, and $\mathbf{u}$ is the set of acceleration commands denoted as

$\mathbf{u} = \begin{bmatrix} u_x & u_y & u_z \end{bmatrix}^T$. This approximation may seem overly simplistic, but the fact that the double integrator dynamics do not depend on heading makes it much easier to simulate vehicle in the cartesian coordinate-based MILP. In cases where the double integrator approximates the vehicle dynamics poorly, it is possible to use more complex plants to estimate vehicles state for the first few steps of the receding horizon plan. One such approach is describes in Section 3.5. Deviations from the double integrator dynamics are also introduced in [44, 37].

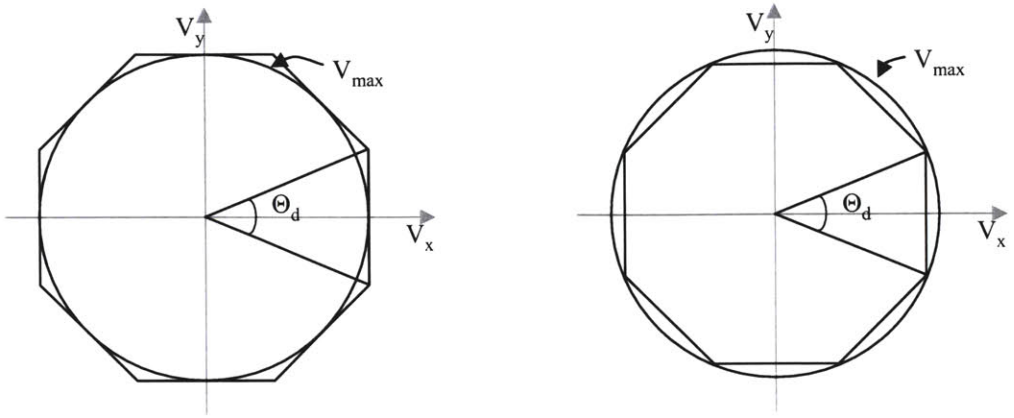
Using a zero-order hold approximation for the changes in state we get:

for $n = 1 \ldots N$

$$\begin{bmatrix} x \\ y \\ z \\ v_x \\ v_y \\ v_z \end{bmatrix}_{n+1} = \begin{bmatrix} 1 & 0 & 0 & \Delta t_n & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t_n & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t_n \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ v_x \\ v_y \\ v_z \end{bmatrix}_n + \begin{bmatrix} \Delta t_n^2/2 & 0 & 0 \\ 0 & \Delta t_n^2/2 & 0 \\ 0 & 0 & \Delta t_n^2/2 \\ \Delta t_n & 0 & 0 \\ 0 & \Delta t_n & 0 \\ 0 & 0 & \Delta t_n \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix}_n \tag{2.9}$$

2.3.2 Velocity and Acceleration Constraints

The acceleration and velocity capabilities of the vehicle must be built into the MILP problem so that the result is dynamically feasible. This is accomplished by putting limits on the velocity and acceleration. The maximum speed and acceleration in the $x - y$ plane are called $V_{\max}$ and $U_{\max}$, respectively. The maximum speed and acceleration in the z direction are called $V_{z\,\max}$ and $U_{z\,\max}$. These constraints cannot be coded into the MILP directly, because they are not linear. The velocity circles are linearized by approximating them as D-sided polygons, as displayed in two dimensions in Figure 2-6. The MILP constrains the velocity vector to stay inside the polygon. The resulting 2D velocity constraints are:



(a) The over-estimating case

(b) The under-estimating case

Figure 2-6: The velocity bounding constraints should be scaled such that the max velocity is never exceeded. A D of 8 is used in the above figures.

$$\text{for } d = 1, \ldots, D$$

$$\Theta_d = \frac{2\pi d}{D} \tag{2.10}$$

$$V_{\max} \geq \cos\Theta_d v_x + \sin\Theta_d v_y \tag{2.11}$$

The above equations will allow for speeds greater than V_{max}, since the D-sided polygon is drawn *around* a circle with radius $V_{\max}$, as shown in Figure 2-6(a). To avoid this overestimation, the D-sided polygon has to be sized so that it is *inside* a circle with radius $V_{\max}$, as shown in Figure 2-6(b). This is achieved by substituting $V_{\max}$ for $V'_{\max}$ into Equation 2.11 where:

$$V'_{\max} = V_{\max}\cos(\pi/D) \tag{2.12}$$

In three dimensions, the bounds on velocity go from being a discretized circle to a 3D surface. Many three dimensional MILP formulations have opted to leave the $x-y$ and z velocity and acceleration bounds decoupled, but here we will introduce another method. Since we know the maximum speed and the maximum climb/descend rate, the velocity constraints will be constructed as three circles, concentric in $x-y$ plane

and located at $\{-V_{z\ \max}, 0, V_{z\ \max}\}$ in the z plane. The radius of the circle at $z = 0$ is simply $V'_{\max}$ and the radii of the circles at $\pm V_{z\ \max}$ are calculated using the velocity triangle pictured in Figure 2-7 as:

$$R_{z\ \max} = \sqrt{V_{\max} - V_{z\ \max}} \cos(\pi/D) \tag{2.13}$$

The elevation angle of the circles at $\pm V_{z\ \max}$ is:

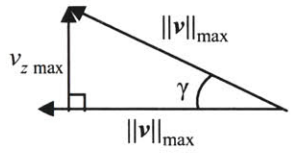


Figure 2-7: Velocity triangle used to construct the 3D velocity constraints.

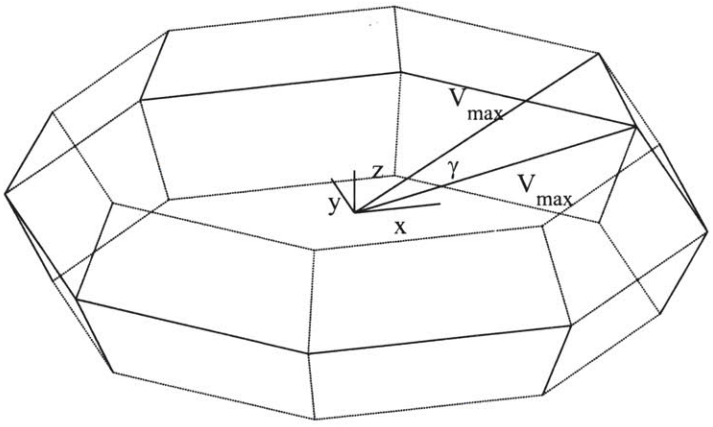


Figure 2-8: Representation of the velocity constraints in three dimensions.

$$\gamma = \sin^{-1}\left(\frac{V_{z\ \max}}{V_{\max}}\right) \tag{2.14}$$

The constraints for velocity bounding in three dimensions are:

for $n = 1 \ldots N, d = 1 \ldots D$

$$V'_{\max} \geq \cos\Theta_d v_{xn} \sin\gamma + \sin\Theta_d v_{yn} \sin\gamma + v_{zn} \sin -\gamma$$
$$V'_{\max} \geq \cos\Theta_d v_{yn} + \sin\Theta_d v_{zn}$$
$$V'_{\max} \geq \cos\Theta_d v_{zn} \sin\gamma + \sin\Theta_d v_{xn} \sin\gamma + v_{zn} \sin\gamma \quad (2.15)$$

Another variation is to add another set of variables, V_n, to the problem that approximates the magnitude of velocity at each time step n. V_n is bounded on $[0,V'_{\max}]$. This variable is useful because obtaining the magnitude of velocity requires a two-norm approximation (discussed later in this chapter), so by combining the velocity bounding with the two-norm approximation, the number of constraints in the problem is decreased. It is much more straightforward to construct "min fuel" and "min speed" type objectives when magnitude variables are directly constructed in the problem formulation. Previous MILP implementations have done this with two sets of constraints, but here we will accomplish the same effect with one set of constraints. The equations for velocity bounding are now:

for $n = 1 \ldots N, d = 1 \ldots D$

$$V'_{\max} \geq \cos\Theta_d v_{xn} \sin(\gamma) + \sin\Theta_d v_{yn} \sin(\gamma) + v_{zn} \sin -\gamma$$
$$V'_{\max} \geq \cos\Theta_d v_{yn} + \sin\Theta_d v_{zn}$$
$$V'_{\max} \geq \cos\Theta_d v_{zn} \sin(\gamma) + \sin\Theta_d v_{xn} \sin(\gamma) + v_{zn} \sin\gamma \quad (2.16)$$

The accuracy of the approximation of the magnitude of velocity is based on the level of discretization, D. Table 2.3.2 shows the level of accuracy for several values of D.

Substituting $u_{\{x,y,z\}}$ for $v_{\{x,y,z\}}$ in the above equations gives the constraints on vehicle acceleration. D_u represents the discretization level for acceleration. The acceleration constraints are:

Table 2.1: Minimum accuracy for the magnitude of velocity or acceleration.

D	Accuracy
4	70.7%
8	92.4%
16	98.0%
32	99.5%

for $d = 1 \ldots D_u$

$$\Theta_d = \frac{2\pi d}{D_u} \tag{2.17}$$

$$\alpha = \sin^{-1}\left(\frac{u_{z\,\max}}{\|u_{\max}\|}\right) \tag{2.18}$$

for $n = 1 \ldots N, d = 1 \ldots D_u$

$$\begin{aligned} U_n &\geq \cos\Theta_d u_{xn} \sin -\alpha + \sin\Theta_d u_{yn} \sin -\alpha + u_{zn} \sin -\alpha \\ U_n &\geq \cos\Theta_d u_{yn} + \sin\Theta_d u_{zn} \\ U_n &\geq \cos\Theta_d u_{zn} \sin\alpha + \sin\Theta_d u_{xn} \sin\alpha + u_{zn} \sin\alpha \end{aligned} \tag{2.19}$$

Changes in Acceleration

An additional constraint on acceleration is required on some situations. If the vehicle dynamics are slow compared to the time step size, then the vehicle may not be able to accelerate from $U_{\min}$ to $U_{\max}$ in time Δt. This need for this additional dynamic constraint was recognized when using very fast re-plan rates. The equations derived here will ensure that dynamically infeasible acceleration commands do not appear in the MILP solution. The change in acceleration between time steps must be limited with a set of constraints. We will define the system parameter $\Delta U_{\max}$ which corresponds to the maximum that the vehicle acceleration U can change per unit of time (nominally seconds). If the minimum time step size $\Delta t_{\max}$ multiplied by $\Delta U_{\max}$ is greater than the maximum change in acceleration possible allowed in the magnitude

constraints above (i.e., going from $-U_{\max}$ to $U_{\max}$ will equal a change of $2U_{\max}$), then no additional constraints are required in the problem. If this is not the case, then constraints must be added for the time steps where the test fails.

$$\text{if } \Delta t_n \, \Delta U_{\max} \leq 2\, U_{\max}, \;\; n = 1, \ldots, N$$

then $\mathbf{u}$ is also subject to

$$\begin{aligned} u_{xn} - u_{x(n-1)} &\leq \Delta t_n \, \Delta U_{\max} \\ -u_{xn} + u_{x(n-1)} &\geq \Delta t_n \, \Delta U_{\max} \end{aligned} \tag{2.20}$$

2.4 Obstacle Avoidance

2.4.1 Obstacle Englargement

In this thesis we will limit ourselves to considering box shaped obstacles. Other papers have covered the formulation for generic obstacle shapes [44], and the improvements suggested here can easily be ported to those techniques. Every obstacle is expanded by two different factors before it is entered into the MILP as a constraint. First, the size of the vehicle needs to be added to the obstacle. Adding this factor to the obstacle creates the *configuration space*, the set of all point where the vehicle is safe (see Figure 2-9). This allows us to treat the vehicle as a point mass.

MILP solutions tend to lie on constraint boundaries, so a second factor needs to be added to the obstacle bounds. If two MILP waypoints lie on the edge of the configuration space around an obstalce, then the straight line path between those points will lie in an unsafe area, as is apparent in Figure 2-10. To prevent this unsafe situation from occuring, the obstacle's dimensions must be expanded in the MILP framework, so that collision with the actual obstacle is made impossible. The amount of expansion required is derived in [25] as:

$$d_{\text{safe}} = \Delta t V_{max} \frac{\sqrt{2}}{4} \tag{2.21}$$

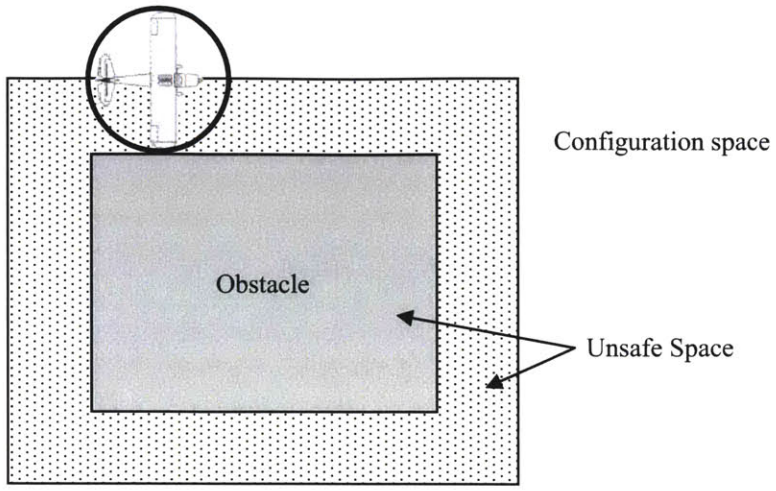


Figure 2-9: Adding vehicle bounds to the obstacle in order to create the configuration space

The method can produce overly conservative trajectories when the time steps or V_{max} is large. An improvement on this approach (in accordance with the *variable MILP* paradigm) is to allow the obstacle expansion to vary each time step. In a variable time step scenario, you have smaller Δt's for the closer time steps, and large ones for the more distant time steps. The expansion now becomes:

$$d_{\text{safe}\,n} = \Delta t_n V_{\max} \frac{\sqrt{2}}{4} \tag{2.22}$$

The calculation can be refined even more by addressing the fact that $V_{\max}$ is not a good upperbound estimate for velocity for each time step n. We create a better estimate for the maximum velocity at each timestep using knowledge of the vehicle's maximum acceleration (Figure 2-12). Given the initial velocity vector $\mathbf{v}_0 = \begin{bmatrix} v_{x0} & v_{y0} & v_{z0} \end{bmatrix}^T$, the upperbound and lowerbound on velocity timestep n will be:

$$\mathbf{v}_{n\,\text{sup}} = \mathbf{v}_0 + \sum_{n=1}^{N} \Delta t_n \, \mathbf{u}_{\max} \tag{2.23}$$

$$\mathbf{v}_{n\,\text{inf}} = \mathbf{v}_0 - \sum_{n=1}^{N} \Delta t_n \, \mathbf{u}_{\max} \tag{2.24}$$

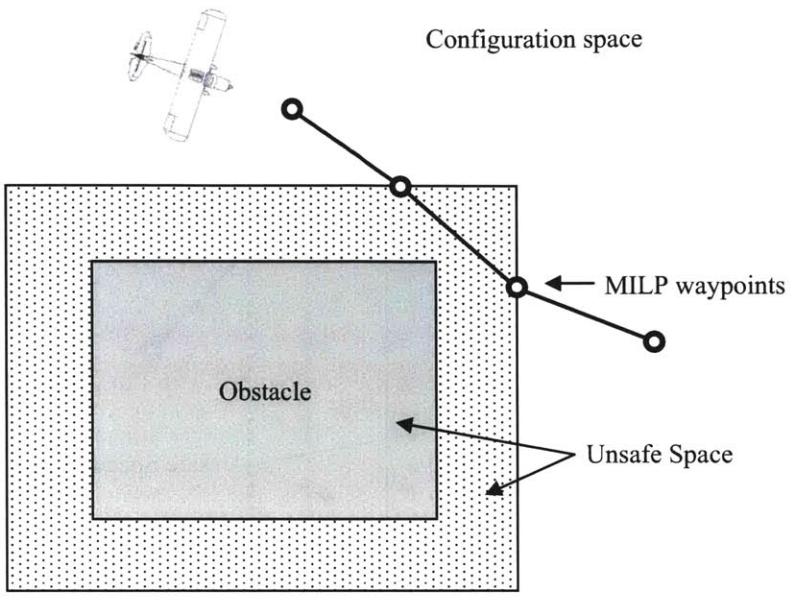


Figure 2-10: Even though the MILP points are outside of the obstacle, the trajectory is not safe.

subject to the constraint that the x and y velocity components at each time step must be less than the maximum speed and the z component must be less than the maximum climb/descend rate.

Using the derivations for $\mathbf{v}_{n\,\mathrm{sup}}$ and $\mathbf{v}_{n\,\mathrm{inf}}$ derived in Equations 2.23 and 2.24, we can restate the required obstacle expansion:

$$\begin{aligned} \|V_n\|_{\mathrm{sup}} &= \max \|v_{xn\{\mathrm{sup,inf}\}}, v_{yn\{\mathrm{sup,inf}\}}, v_{zn\{\mathrm{sup,inf}\}}\| \\ d_{\mathrm{safe}\,n} &= \Delta t_n \, \|V_n\|_{\mathrm{sup}} \, \frac{\sqrt{2}}{4} \end{aligned} \tag{2.25}$$

By combining the smaller Δt's available when using variable time stepping with the new upper bound on velocity at each time step, we can significantly limit the amount the obstacle expansion necessary, especially for the closer time steps, without decreasing safety at all.

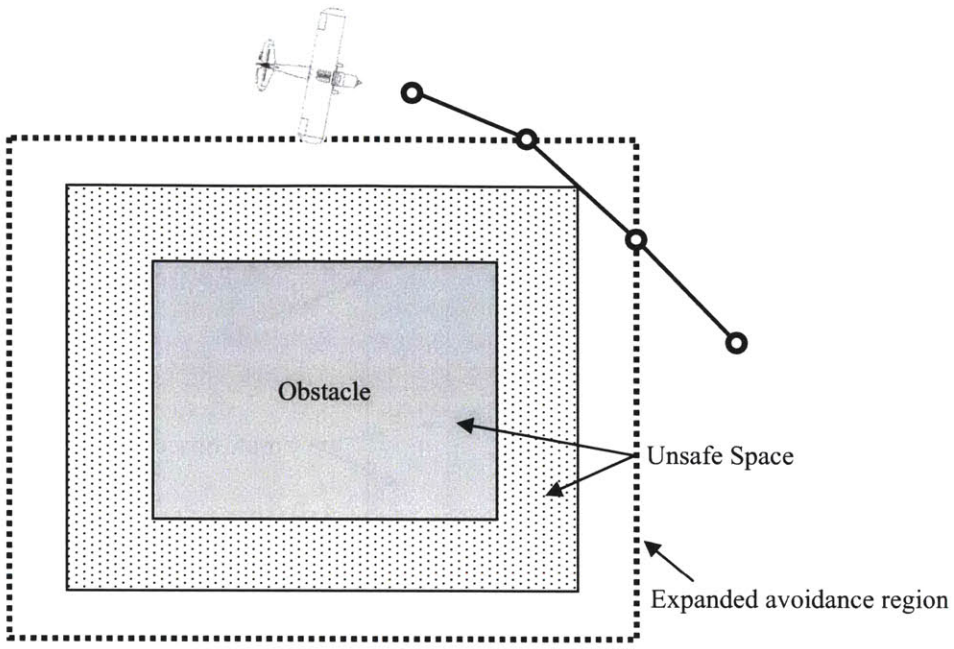


Figure 2-11: Expanding the bounds of the obstacle to avoid the possibility of collision

2.4.2 Adding the Obstacle Constraints

Now that the proper bounds for the obstacle have been calculated, we can can add the actual constraint to the MILP. We will define the expanded obstacles using the position of its upper and lower bounds in each dimension: $x_l, x_u, y_l, y_u, z_l, z_u$. The constraints that needed to be added at each time step are:

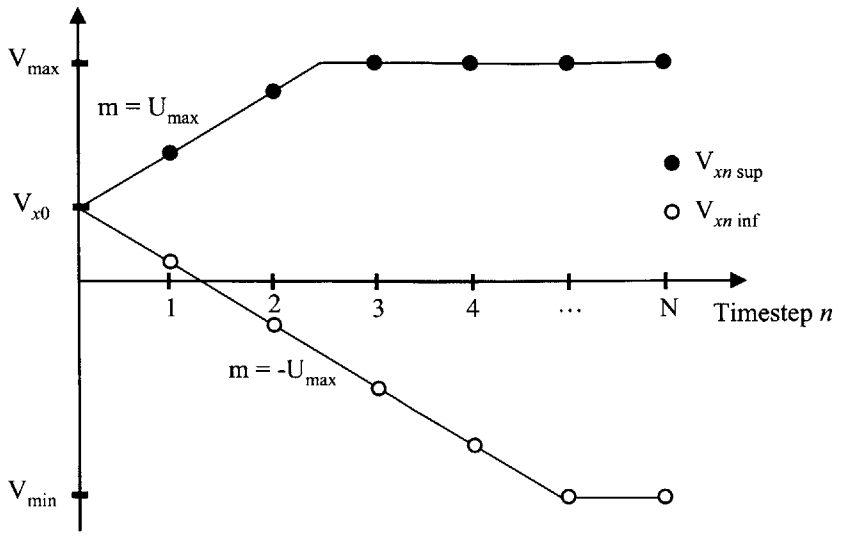


Figure 2-12: Graph showing how $\mathbf{v}_{n\,\mathrm{inf}}$ and $\mathbf{v}_{n\,\mathrm{sup}}$ change with time. The x direction is shown. The other two dimensions would look very similar.

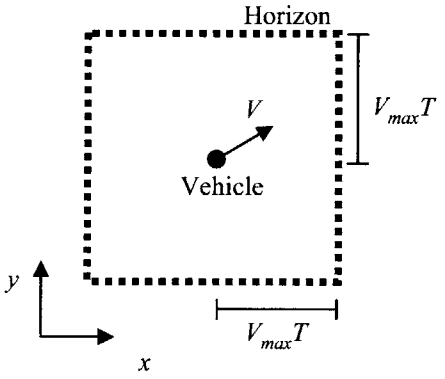
$$
\begin{aligned}
\text{for } n = 1 \ldots N & \\
x_n &\leq x_{l\,n} - M\, b_{n1} \\
x_n &\geq x_{u\,n} + M\, b_{n2} \\
y_n &\leq y_{l\,n} - M\, b_{n3} \\
y_n &\geq y_{u\,n} + M\, b_{n4} \\
z_n &\leq z_{l\,n} - M\, b_{n5} \\
z_n &\geq z_{u\,n} + M\, b_{n6} \\
\sum_{i=1}^{6} b_{ni} &\leq 5
\end{aligned}
\tag{2.26}
$$

The variable M in the equations above is a suitably large number that forces the binary variable b_{ni} to turn on when the equality would not otherwise be met. This so-called "big M" formulation is used widely in mixed integer linear programming, and is explained thoroughly in [1]. The binaries and M are used to create a hard constraint in the MILP. By never allowing the sum of all the binaries to be 6, the location $\mathbf{x}_n$ will never be inside of the obstacle bounds.

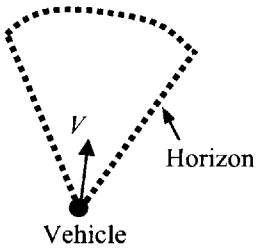
2.5 Horizon minimization

This section introduces a new algorithm for determining a lower bound on the *planning horizon* for the MILP problem. The planning horizon is the set of reachable position states each time step. This new method guarantees safety, in that the vehicle will not travel outside of the planning horizon and collide with obstacle left out of the MILP. This process of removing unreachable obstacles from the MILP can reduce problem complexity drastically compared to some of the previous planning horizon implementations [25].

In the past, MILP path planning implementations have used two types of horizons. The first method sized the horizon by creating a box with side length $2\,V_{\max}\,T_s$, where T_s is the MILP simulation time [25]. This was intended as a way to include all obstacles reachable in time T_s. In most cases, though, this method tends to create a horizon that is much too large.



(a) The horizon is a box with a size based on the maximum velocity and the simulation time [25]. The horizon is much larger than it needs to be.



(b) The horizon is a cone centered along the velocity vector [44]. Reachable object could potentially be ignored.

Figure 2-13: Two types of horizons used in previous MILP formulations

The second method has been to formulate the horizon as a cone in front of the vehicle [44]. The intention here was to minimize the number of obstacles that have to be included at each solve iteration. If the vehicle trajectory is not constrained to stay in the cone, then the MILP result could command the vehicle to fly outside of the cone and into obstacles that were left out of the problem.

By creating a better lower bound on the set of reachable position states at each timestep, all the relevant obstacles are still included in the problem, but the unnecessary obstacles are not. At timestep $n = 0$ the planning horizon is one point, the initial position. As n increments, the planning horizon expands. A unique planning horizon is created at each time step. The planning horizon is calculated using the initial conditions before the MILP is updated between solve iterations.

The expansion of the horizon at each timestep is based on the maximum acceleration and velocity of the vehicle. Integrating the maximum velocity estimate Equations 2.23 and 2.24 gives the maximum position states reachable at each timestep. These estimates of $\mathbf{x}_{n\,\mathrm{inf}}$ and $\mathbf{x}_{n\,\mathrm{sup}}$ will define the horizon size.

$$\mathbf{x}_{n\,\mathrm{sup}} = \mathbf{x}_0 + \sum_{n=1}^{N} \Delta t_n \, \mathbf{v}_{n\,\mathrm{sup}} \tag{2.27}$$

$$\mathbf{x}_{n\,\mathrm{inf}} = \mathbf{x}_0 - \sum_{n=1}^{N} \Delta t_n \, \mathbf{v}_{n\,\mathrm{inf}} \tag{2.28}$$

The worst-case horizon size using the new method described is still smaller than the previous cube planning horizon sized by $\mathbf{v}_{\max}$ and T_s. The greatest improvement is when the vehicle has a large speed range in proportion to $U_{\max}$. Keeping the horizon as small as possible while maintaining safety is advantageous because in doing so you can avoid having to add unnecessary obstacles into the MILP formulation. Even though the obstacle are unreachable, they still contribute to the complexity of the MILP and slow down the solution time.

To illustrate the importance of minimizing the horizon space, two simple simulation are presented. The air vehicle flies from from one point to another. There is one obstacle in the middle of the environment, and 20 obstacles around the edge of the environment. The first simulation uses the horizon sized as a cube with side length $2\,V_{\max}\,T_s$, as used in several previous MILP implementations. Using this method, all of the obstacles are included in the MILP problem. Figure 2-15 shows the environment, vehicle state, and MILP plan after nine solve iterations. The second simulation

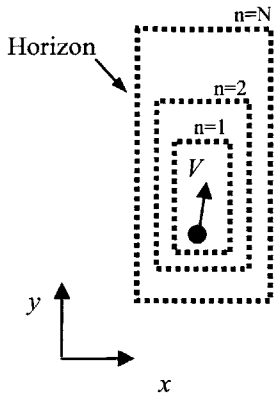


Figure 2-14: New planning horizon formulation uses the initial velocity to calculate a small, but still safe, horizon that varies with each time step.

uses the same environment, but applies the horizon minimization method described above. On average only about 2 obstacles are included in the MILP problem at each solve iteration. The first simulation was computed with an average solve time of 66.6ms, and the second simulation was computed with a mean time of 31.8ms. This is a considerable difference when you consider the fact that the majority of the extra obstacles included in the first formulation were not even in a reachable space at each time step.

Table 2.2: MILP problem characteristics using different horizon sizes.

Test Type	Mean Solve Time	Mean # of Variables	Mean # of Constraints
Fixed Horizon	$66.66ms$	483	967
Variable Horizon	$31.83ms$	153	582

2.6 Two-norm Approximation

There a many situations in a MILP path planning formulation where it is useful, or even necessary, to obtain an estimate of the magnitude of a vector. Unfortunately, the process of taking the two-norm of a vector, even a one-dimensional one, is nonlinear. It is necessary to approximate magnitudes as piece-wise linear constraints within the MILP framework. The two-norm approximations described in this section are key

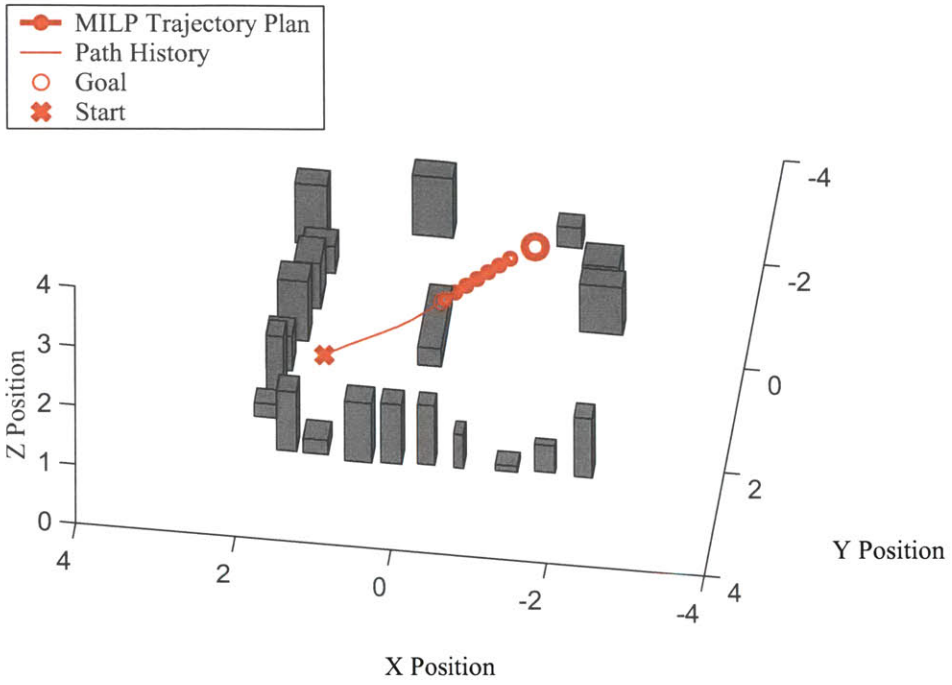


Figure 2-15: Simulation where the vehicle is in an obstacle-rich environment, but the obstacles are outside of the reachable space.

techniques used frequently throughout this thesis. The two-dimension case has been introduced before, but the one and three dimension cases are new.

2.6.1 In One Dimensions

In one dimension, the two-norm is simply the absolute value. A variable F is defined in the MILP that will hold the value $\|f\|$. Because F is an absolute value, it is only define on the bounds $[0, \infty)$. If the two-norm of f will be minimized in the objective function, then the convex set of constraints is:

$$\begin{aligned} f &\leq F \\ -f &\leq F \end{aligned} \tag{2.29}$$

If the two-norm of f will be maximized in the objective function, then constraints

become non-convex. Binary variable are used to ensure that only one constraint is true:

$$\begin{aligned} f &\geq F - M(1 - b_i) \\ -f &\geq F - M(1 - b_i) \\ \sum_{i=1}^{2} b_i &= 1 \end{aligned} \tag{2.30}$$

Depending on the capabilities of the solver, the convex and non-convex constraints can be combined using logical or indicator constraints (see Section 5.2.2).

2.6.2 In Two Dimensions

The two-norm approximation for estimating distance is very similar to the process by which velocity and acceleration is bounded, as described in Section 2.3.2.

The position of the vehicle at time step n is $\mathbf{x}_n = (x_n, y_n, z_n)$. The reference point is located at $\mathbf{r} = (x_r, y_r, z_r)$. The distance between the vehicle and the reference point at some time step n is $\|\mathbf{x}_n - \mathbf{r}\|$.

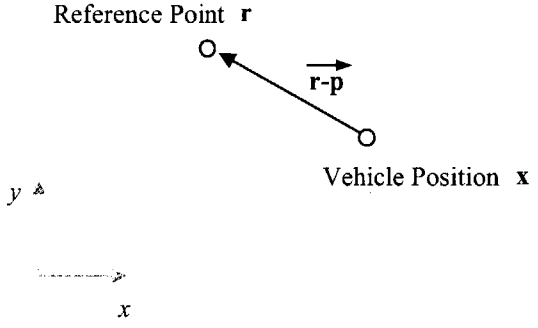


Figure 2-16: Vector from vehicle to the reference point

We know that $\sqrt{(r_x - x_n)^2 + (r_y - y_n)^2} = \|\mathbf{r} - \mathbf{x}_n\|$. Graphically, this appears as a circle with radius $\|\mathbf{r} - \mathbf{x}_n\|$ centered at the origin. We will discretize the circle as a D-sided polygon and define magnitude approximation variable, R_n, at each timestep. When R_n is minimized in the objective function, the constraints are convex:

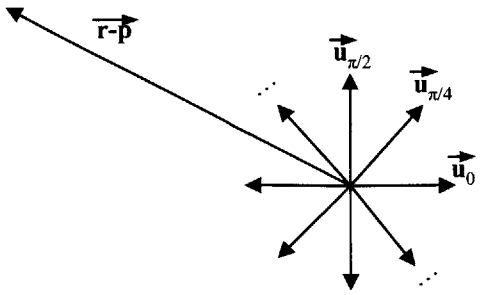


Figure 2-17: The unit circle is discretized as a set of equally spaced unit vectors and dotted with the $\mathbf{x}_n - \mathbf{r}$ vector

$$\text{for } d = 1, \ldots, D$$

$$\Theta_d = \frac{2\pi d}{D} \tag{2.31}$$

$$R_n \geq (r_x - x_n)\cos(\theta_d) + (r_y - y_n)\sin(\theta_d) \tag{2.32}$$

These constraints effectively assign the value of R_n to be the maximum value of $(r_x - x_n)\cos\theta_d + (r_y - y_n)\sin\theta_d$ at each of the discretization points. When the intention is to maximize R_n, the constraints become non-convex, just like in the one dimension case. Again binary variables are used to ensure that only one of the constraints is active. The constraints for the maximizing the two-norm are:

$$\text{for } d = 1 \ldots D$$

$$R_n \leq (r_x - x_n)\cos(\theta_d) + (r_y - y_n)\sin(\theta_d) + M(1 - b_i) \tag{2.33}$$

$$\sum_{i=1}^{D} b_i = 1 \tag{2.34}$$

2.6.3 In Three Dimensions

The process of finding the two-norm for a vector with three dimension is very similar to the 2-D case, but simply adding another dimensions to the two-dimensional two-norm approximation is not reasonable. Linearizing the constraints about a sphere requires

D^2 constraints, compared to only D constraints in the 2D case. For a scenario with 10 time steps and a discretization $D = 10$, this would result in 580 more constraints! For this reason we will look at two approaches that minimize the number of constraints needed to represent the three dimensional two-norm approximation.

The first approach limits the maximum speed attainable in any of the three axes planes $x - y$, $x - z$, and $y - z$. The number of constraints required for this method is $3D$.

$$\begin{aligned} &\text{for } d = 1, \ldots, D \\ &\qquad R_n \geq (x_r - x_n)\cos\theta_d + (y_r - y_n)\sin\theta_d \\ &\qquad R_n \geq (y_r - y_n)\cos\theta_d + (z_r - z_n)\sin\theta_d \\ &\qquad R_n \geq (z_r - z_n)\cos\theta_d + (z_r - z_n)\sin\theta_d \end{aligned} \tag{2.35}$$

The second approach is the one used to limit velocity and acceleration. This method also produces $3D$ constraints, and uses three concentric circles to bound the magnitude approximation.

$$\begin{aligned} &\text{for } n = 1, \ldots, N, d = 1, \ldots, D_u \\ &\quad U_n \geq (x_r - x_n)\cos\Theta_d \sin\alpha + (y_r - y_n)\sin\Theta_d \sin\alpha + (z_r - z_n)\sin\alpha \\ &\quad U_n \geq (x_r - x_n)\cos\Theta_d + (y_r - y_n)\sin\Theta_d \\ &\quad U_n \geq (x_r - x_n)\cos\Theta_d \sin -\alpha + (y_r - y_n)\sin\Theta_d \sin -\alpha + (z_r - z_n)\sin -\alpha \end{aligned} \tag{2.36}$$

2.7 The Objective Function

The objective function is a critical component of any mathematical programming problem. The selection of variables and weights used in the objective determines the result of the MILP. Throughout this thesis we will minimize the objective function, so that variables with a negative weight will be maximized and those with a positive

weight will be minimized. All of the demonstrations in this thesis include a minimization on acceleration at each time step. This penalty on acceleration, however slight, will keep the resulting trajectory smooth. We will define the foundation objective function $\mathcal{O}$ as:

$$\mathcal{O} = \sum_{n=1}^{N} U_n \tag{2.37}$$

A trajectory with sharp acceleration changes is inefficient, and the vehicle will likely have a hard time following the trajectory. We will modify and append new variable to the objective function as the new features are introduced.

2.8 Waypoint Handling

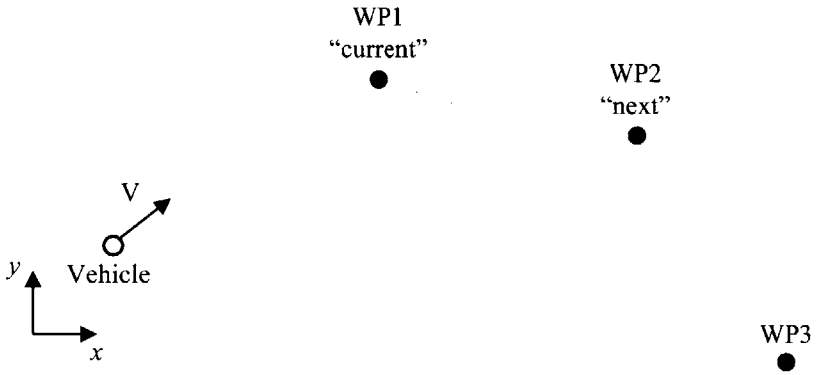


Figure 2-18: Labeling for mission planning waypoints

For the sake of clarity, we will define "mission planning waypoints" as waypoints provided from outside the MILP framework, while "MILP waypoints" are the waypoints that come from the MILP solution and are used to guide the vehicle.

Waypoints are generally looked at as points in space that must be passed through. All mission planning waypoints in the MILP framework are implemented as regions of space centered around what would be considered the "true" waypoint. There are two primary reasons for doing this. First, requiring the vehicle state to meet certain values with 32 bit precision puts an enormous burden on the solver, and will often

result in an infeasible solution. Second, the discrete nature of the MILP time steps means that the vehicle could be forced to unnecessarily slow down or speed up so that one of the MILP waypoints lays on top of the mission planning waypoint.

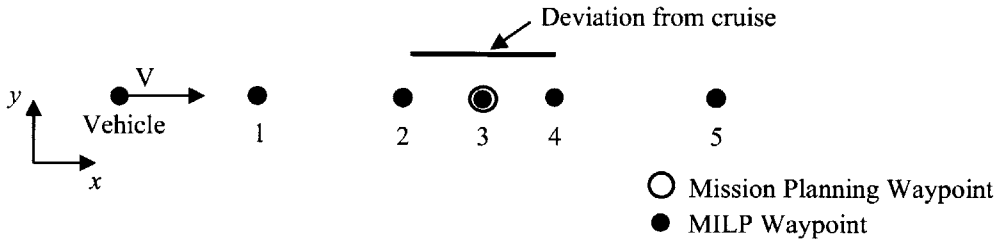


Figure 2-19: The vehicle must slow down so that one of the MILP waypoints matches with the mission planning waypoint

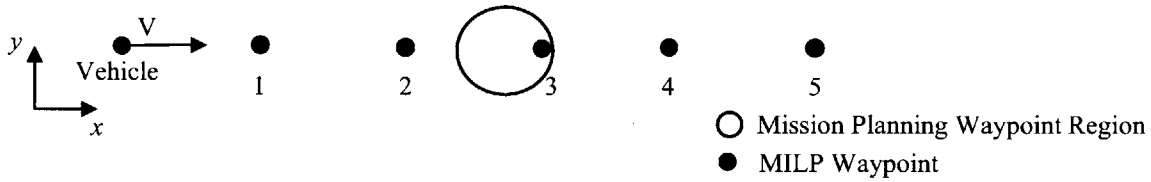


Figure 2-20: Expanding the mission waypoint helps prevent unnecessary accelerations

2.8.1 Waypoint expansion

Waypoint expansion relaxes the constraint of having to pass through a particular point in space. Instead, an entire region around the waypoint is made acceptable. In the past, these regions were very small, acting simply as a $\pm\delta$ regions that would ease computational complexity. This section introduces a more formal way of expanding the waypoint region size. The typical result is smoother trajectories with a lower objective cost.

Maximum expansion is required when the vehicle is approaching the waypoint at $V_{\max}$, and the mission planning waypoint lies in the middle of two MILP waypoints. During the time that the vehicle is between time steps t_n and t_{n+1}, it will travel a distance $\Delta t_g V_{\max}$. Thus, by expanding the mission planning waypoint to encircle a

circular region in the x-y plane with radius $0.5\Delta t_g V_{\max}$ we can guarantee that one of the MILP waypoints will be included in the region, without requiring the vehicle to slow down.

2.9 Summary

In this chapter we have introduced the principle components used in a MILP path planner. State equations and bounds on velocity and acceleration are used to account for vehicle dynamics in the optimization problem. A new obstacle expansion approach uses a better estimate of the maximum velocity at each time step to minimize the level of expansion required. Horizon minimization reduces the number of obstacles in the MILP at each time step. The extensions introduced here have an overall effect of reducing solve time and make the resulting trajectories less conservative.

Chapter 3

Extended MILP Capabilities

3.1 Introduction

The features covered in this chapter can be added into the core formulation, described in Chapter 2, to increase the functionality of a MILP path planner. Some techniques have appeared in previous MILP path planning papers, and others include original modifications designed to increase functionality or decrease complexity. The modifications introduced in this chapter, such as the use of linear interpolations points, allow for more advanced trajectories in a real flight environment.

3.2 Linear Interpolation Points

The variable time step approach described in Chapter 2 is useful because it minimizes the amount of decision variables in the problem, by allowing a more coarse plan for the distant waypoints. The shortcoming is that the more distant obstacles must be expanded to a greater degree, since Δt_n is larger as n increases. One way to limit this obstacle expansion is by adding linear interpolation points in between the MILP waypoints. A linear interpolation point uses the decision variables obtained at time t_n to estimate the vehicle state at some pre-determined time t_s where $t_n \leq t_s \leq t_{n+1}$ for the sake of collision avoidance checks. Multiple linear interpolation points can be placed between two MILP waypoints. This effectively breaks Δt_n into two pieces,

where $\Delta t_{n1} = t_s - t_n$ and $\Delta t_{n2} = t_{n+1} - t_s$. The amount of obstacle expansion required is decreased because now the smaller Δt values are used in the expansion calculatation. Minimizing Δt in this fashion is less computationally expensive than adding more MILP points to the problem, but still keeps the formulation from getting too conservative.

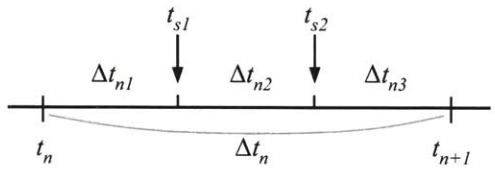


Figure 3-1: Two linear interpolation points, located at t_{s1} and t_{s2}, are placed between MILP waypoints.

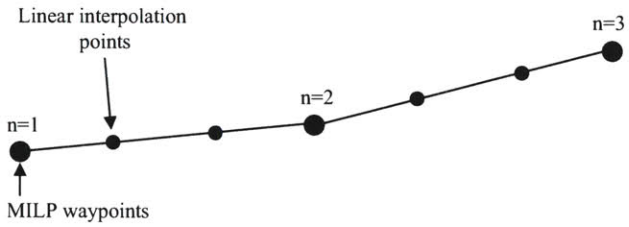


Figure 3-2: Linear interpolation points are placed between MILP waypoints.

Given two MILP waypoints $\mathbf{x}_{n-1}$ and $\mathbf{x}_n$ with spacing Δt_n and G linear interpolation points between the equation that define the set of linear interpolation point L, where $\mathbf{L}_i = [L_{xi}, L_{yi}, L_{zi},]^T, i \in G$.

$$\text{for } n = 1 \ldots N, i = 1 \ldots G$$
$$\mathbf{L}_i = \left(1 - \frac{i\ \Delta t_n}{G+1}\right)\ \mathbf{x}_{n-1} + \frac{i\ \Delta t_n}{G+1}\ \mathbf{x}_n \tag{3.1}$$

To illustrate the advantage of using linear interpolation points in obstacle-rich environments, we will look at two simulation. An identical set of tall obstacles is used in both simulations, such that the vehicle cannot fly over the obstacles, it must

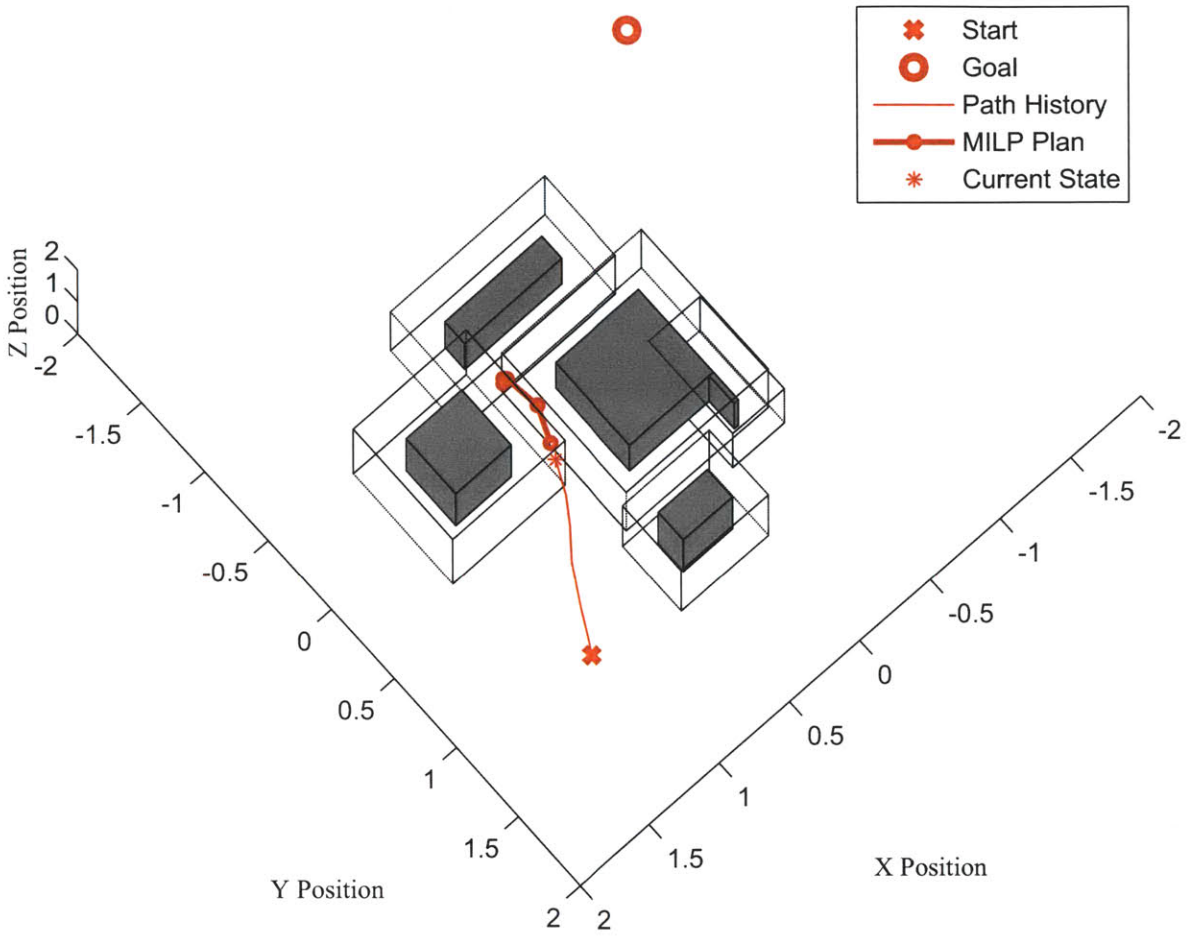


Figure 3-3: Without using linear interpretation points, the magnitude of obstacle expansion causes the vehicle to get stuck.

fly between them to reach the goal state. The objective function is:

$$\text{minimize } \mathcal{O} + \sum_{n=1}^{N} 10\, R_n \tag{3.2}$$

where R_n is the distance approximation to the goal state at each time step and $\mathcal{O}$ is the base objective described in 2.7.

The first simulation, as seen in Figure 3-3 does not use linear interpolation points. The second simulation, displayed in Figure 3-4, does use linear interpolation points. In both cases the initial time step is two seconds, and the other timesteps are spaced at three seconds. One linear interpolation point is placed in between each MILP

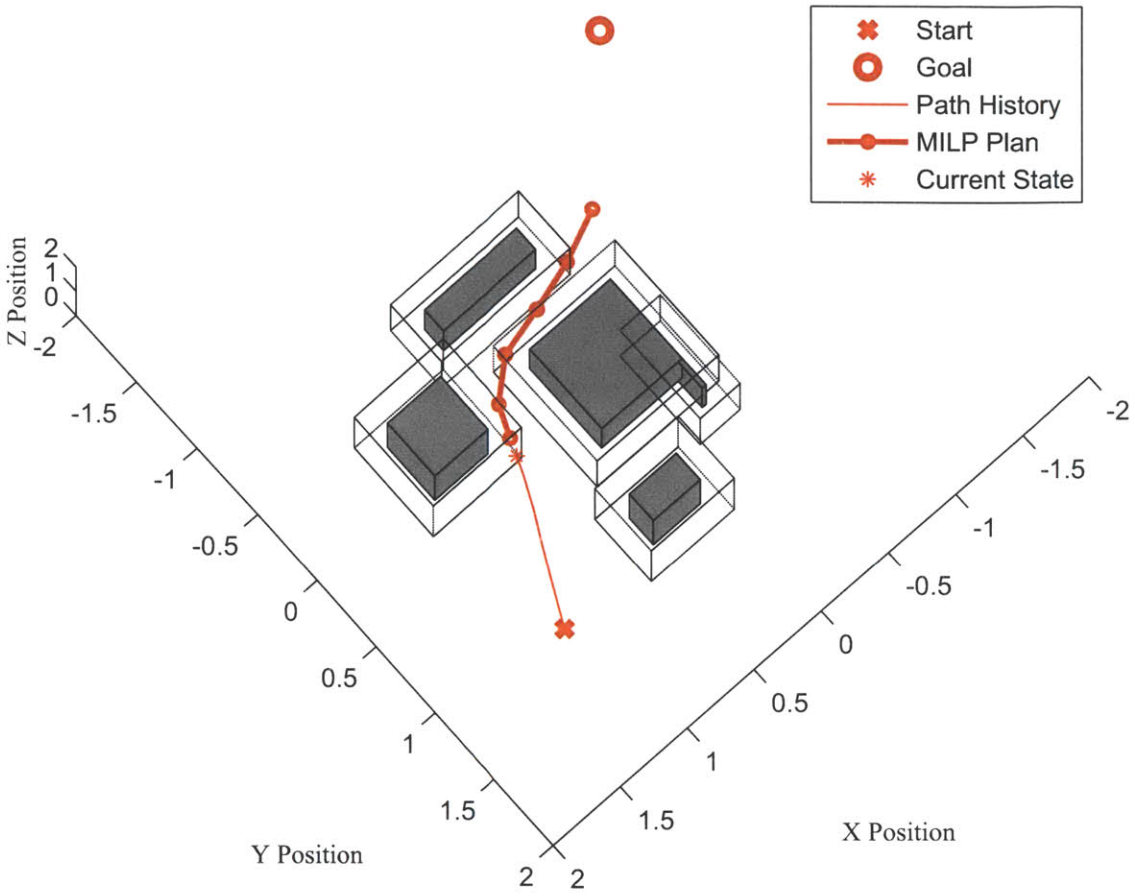


Figure 3-4: Using linear interpretation points, the vehicle is able to reach its goal.

waypoint more than two seconds apart in the second simulation. The dashed lines around the obstacle display the bounds of the the obstacle after it has been expanded according to the techniques in Section 2.4.1. Notice that the vehicle in the first simulation is stuck and will never reach the goal state. The expansion of the obstacles was so great that a potential path to the goal state was blocked. In the second simulation, the level of obstacle expansion is noticeably decreased, and the vehicle was able to reach the goal state successfully. The first seven re-plans are shown for each simulation. The mean solution time for the first seven steps of the simulation without linear interpolation points was 160ms. The corresponding mean solution time with linear interpolation point was 203ms.

3.2.1 Deviation from straight line paths

The MILP framework is designed to ensure that the air vehicle will be safe by checking at intermediate points, such as MILP waypoints and linear interpolation points. Obstacle expansion and other steps are used to guarantee that by checking at these intermediate points, the vehicle will be safe along the line segments connecting each point. But what happens when the vehicle's path deviate from these straight line segments? Here, we will introduce a way to predict deviation and check for safety while staying in a linear framework.

Using knowledge of how an air vehicle will navigate through waypoints, it is possible to move linear interpolation points off of the straight line path and closer to the actual flight path. This allows for more robust collision avoidance checking between MILP points.

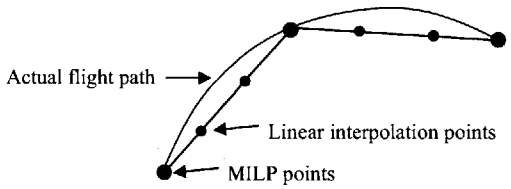


Figure 3-5: The actual flight path can deviate significantly from the linear prediction

There are different ways to estimate deviation from straight line paths, so knowledge of how the outer-loop control system handles the MILP points is critical. For the sake of this example, we will assume that the control system guides that vehicle through the x-y-z coordinates dictated by the MILP solution. We will also assume that the corresponding velocity commands are relaxed for the sake of keeping the trajectory smooth. Different waypoint followers will use different algorithms, so the following equations may need to be modified. The basic principles are the same, nonetheless.

Assuming that there are L linear interpolation points between $\mathbf{x}_n$ and $\mathbf{x}_{n+1}$. We assume that the vehicle passes through every waypoint $\mathbf{x}_n$ tangent to the line con-

necting $\mathbf{x}_{n-1}$ and $\mathbf{x}_{n+1}$. Thus, the actual velocity vector $\mathbf{v}_{n\,\text{actual}}$ is:

$$\mathbf{v}_{n\,\text{actual}} = \frac{\mathbf{x}_{n+1} + \mathbf{x}_{n-1}}{2\Delta t_n} \tag{3.3}$$

The straight line path to the *next* waypoint is combined with the tangent path from the *nearest* waypoint using a weighted average. The resulting locations for the deviation-adjusted linear interpolations points are:

for $l = 1, \ldots, L$

when $l < L/2$

$$\mathbf{l} = \mathbf{x}_n + \frac{l}{2(L+1)}(\mathbf{v}_n + \mathbf{v}_{n\,\text{actual}}) \tag{3.4}$$

when $l = L/2$

$$\mathbf{l} = 1/2(\mathbf{x}_n + 1/2(\mathbf{v}_n + \mathbf{v}_{n\,\text{actual}}) + 1/2(\mathbf{x}_{n+1} - 1/2(\mathbf{v}_n + \mathbf{v}_{n+1\,\text{actual}})) \tag{3.5}$$

when $l > L/2$

$$\mathbf{l} = \mathbf{x}_{n+1} - \frac{l-L}{2(L+1)}(\mathbf{v}_n + \mathbf{v}_{n+1\,\text{actual}}) \tag{3.6}$$

An example result is pictured in two dimensions in Figure 3-6, where $L = 2$. The deviation-adjusted linear interpolation points are clearly able to mimic the aircraft trajectory much better than the straight line segments connecting the MILP waypoints.

3.3 Cost regions

Cost regions are similar to normal obstacles, but with more flexibility. A cost region with a positive value (a *high-cost region*) in the objective function will be avoided and the opposite is the case for regions with a negative cost (*low-cost regions*). Instead of creating hard constraints, such as those described in Section 2.4.2, cost regions are setup so that a cost value (positive or negative) appears in the objective function whenever the MILP solution waypoints pass through a defined area of space. The

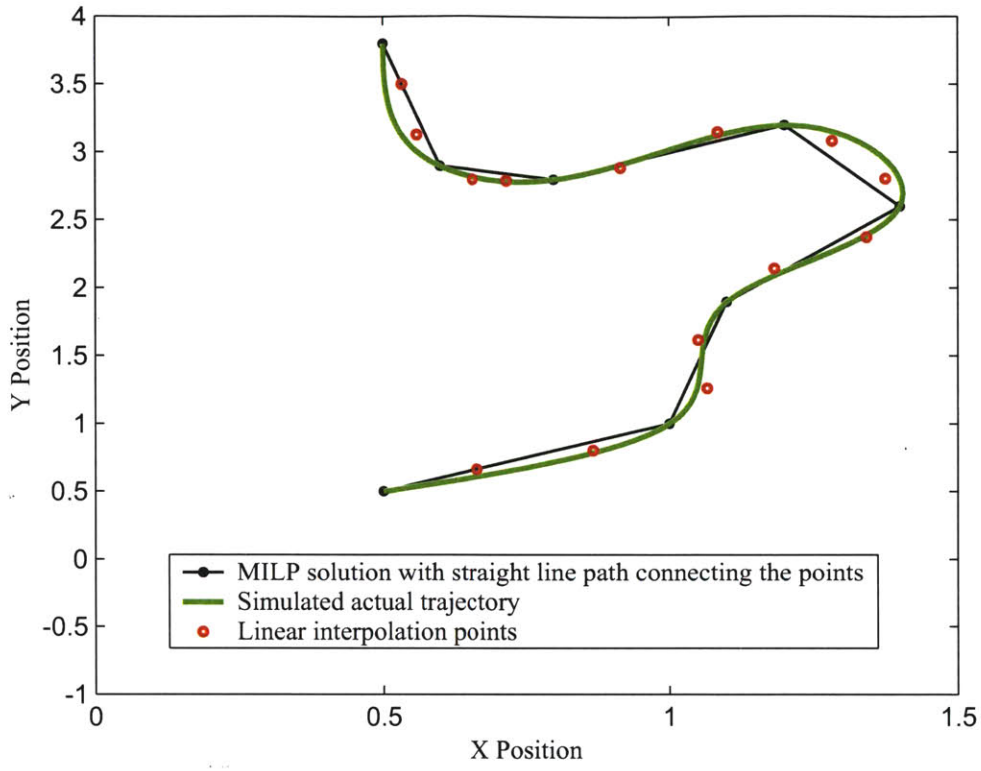


Figure 3-6: The spline curve drawn between the MILP points demonstrates deviation from the straight-line path. The linear interpolation points adjusted to match the actual flight path more closely.

vehicle can fly into very high-cost regions without going infeasible, but at the cost of raising the objective function. Soft constraints have an effect on the objective function, but do not have to be fulfilled. It is sometimes useful to place high cost regions around obstacles so that the vehicle avoids the boundaries of infeasibilty. A rectangular cost region is defined by its lower and upper bounds in each dimension: $x_l, x_u, y_l, y_u, z_l, z_u$. The set of constraints in Equation 3.7 activate the binary b_n when the vehicle state is within the cost region and the cost W_n is added to the objective

function.

$$\begin{aligned}
&\min: \sum_{n=1}^{N} W_n b_n \\
&\text{subject to:} \\
&\qquad x_n \geq x_l + M(b_n - 1) \\
&\qquad x_n \leq x_u - M(b_n - 1) \\
&\qquad y_n \geq y_l + M(b_n - 1) \\
&\qquad y_n \leq y_u - M(b_n - 1) \\
&\qquad z_n \geq z_l + M(b_n - 1) \\
&\qquad z_n \leq z_u - M(b_n - 1)
\end{aligned} \tag{3.7}$$

Using the cost region technique, probability and cost-to-go maps can be added into the MILP formulation with ease by discretizing the 3D as a set of boxes that together take up the entire reachable space and setting the weights on each region appropriately. Cost regions are added to the MILP problem just like regular obstacles; the constraints only need to be added to the problem when the union of the planning horizon and the cost region is not null.

3.4 Moving objects

Obstacle with a non-negligible velocity should not be handled as a static constraint at every time step in the MILP formulation. Here, we will model obstacles with a defined velocity in the MILP formulation so that the region to be avoided is different at each time step. This allows the vehicle to predict and plan around potential future collisions with moving obstacle. Moving obstacles can be other air vehicles, cars, ships, or anything else.

MILP has been used for multi-vehicle trajectory problems in the past. It has been used to maximize spacing between vehicles in [3]. Centralized multi-vehicle path planning has been implemented in [24, 41]. Decentralized planning has also been studied

[38, 46, 27]. The moving obstacle representation can be used to test MILP path planning for multiple vehicles in the environment. The technique described here is an extremely decentralized approach at the multi-vehicle problem. Using only the current state information of the other vehicles, each vehicle navigates through the space while minimizing cost and maintaining safety. This untrusting, non-communicating assumption especially valid when the other vehicle is an adversary. A confidence factor has also be included to deal with the fact that the moving obstacle's velocity will not remain constant over the simulation time. The lower the confidence factor, the more that the avoidance region for the moving obstacle is expanded at each time step.

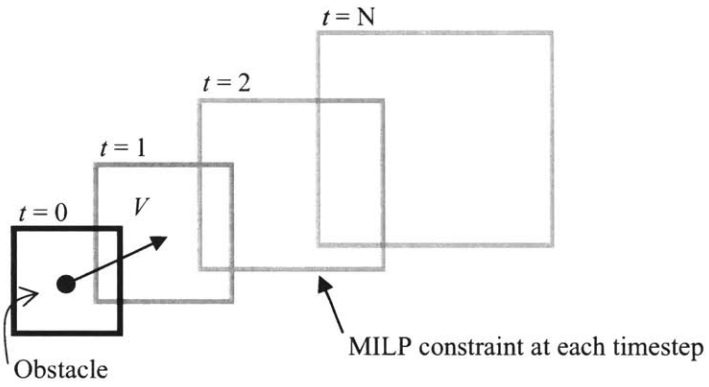


Figure 3-7: The constraint representation for a moving obstacle changes at each time step. The confidence factor of the moving obstacle dictates the rate of expansion between timesteps.

The same initial enlargment strategy as introduced in Section 2.4.1 is used on the moving obstacles. We will assume that the size of the moving obstacle (including the two safety factors) at each time step n is: $x_{ln}, x_{un}, y_{ln}, y_{un}, z_{ln}, z_{un}$. These bounds should be centered about the origin so that $x_{ln}, y_{ln}, z_{ln} < 0$ and $x_{un}, y_{un}, z_{un} > 0$. The moving obstacle has initial position $\mathbf{p} = [p_x, p_y, p_z]^T$ and initial velocity $\mathbf{r} = [r_x, r_y, r_z]^T$ and is given a confidence factor c. The constraints for the moving

obstacles are:

$$
\begin{aligned}
&\text{for } n = 1 \ldots N \\
&\qquad x_n \leq x_{ln}\left(\frac{t_n}{c}\right) + p_x + r_x t_n - M\, b_{n1} \\
&\qquad x_n \geq x_{un}\left(\frac{t_n}{c}\right) + p_x + r_x t_n + M\, b_{n2} \\
&\qquad y_n \leq y_{ln}\left(\frac{t_n}{c}\right) + p_y + r_y t_n - M\, b_{n3} \\
&\qquad y_n \geq y_{un}\left(\frac{t_n}{c}\right) + p_y + r_y t_n + M\, b_{n4} \\
&\qquad z_n \leq z_{ln}\left(\frac{t_n}{c}\right) + p_z + r_z t_n - M\, b_{n5} \\
&\qquad z_n \geq z_{un}\left(\frac{t_n}{c}\right) + p_z + r_z t_n + M\, b_{n6} \\
&\sum_{i=1}^{6} b_{ni} \leq 5
\end{aligned}
\tag{3.8}
$$

The "big M" formulation described in Section 2.4.2 is also used here to create hard constraints. In some situations, it is sensible not make the more distance time steps formulated as hard constraints. By making them soft constraints, infeasible problem formulations are avoided in situations were there is no real danger of collision.

3.5 Using Other Plant Dynamics

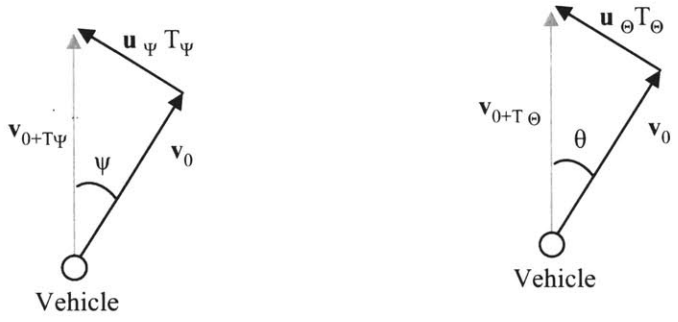
Sometimes approximating the vehicle dynamics as a double integrator produces trajectories that are poorly followed by the vehicle. Other MILP formulation have implemented vehicle dynamics that depart slightly from the double integrator paradigm [44], but the technique introduced in this section can be used for more exotic plants. The drawback is that it is not applied at every timestep. The proper plant dynamics are used for the most immediate planning points in the receding horizon formulation, and double integrator plant dynamics are used for the more distance planning points. The first step is to obtain a linearized plant model for the vehicle at the current state

$\mathbf{S_0} = \begin{bmatrix} \mathbf{x}_0 & \mathbf{v}_0 \end{bmatrix}^T$:

$$\mathbf{x} = A_{S_0}\mathbf{x} + B_{S_0}\mathbf{u} \tag{3.9}$$

Typically, this plant will depend on pitch Θ and heading Ψ, where Θ and Ψ are calculated from the velocity vector $\mathbf{v_0}$. Therefore, the plant is only a good approximation for a range of pitch and heading angles, $\Theta \pm \theta$ and $\Psi \pm \psi$ respectively. θ and ψ are pre-selected based on the strictness required in the modeling of the dynamics. In a real-time scenario, it would be most practical to have a lookup table for the linearized plant, θ, and ψ based on S_0 rather than have to calculate it between iterations.

It is necessary to determine at which time steps the new plant dynamics can be applied. In other words, how long will the vehicle stay within $\Theta \pm \theta$ and $\Psi \pm \psi$? At time step n, the total simulation time $T(n)$ is $\sum_{i=1}^{n} \Delta t(n)$. Assuming an initial acceleration called $\mathbf{u}_\Psi$ where $\|\mathbf{u}_\Psi\| = u_{\max}$ and $\mathbf{u}_\Psi$ is orthogonal to the velocity vector $\mathbf{v}$ and parallel to the ground plane will induce a maximum $\frac{\partial \Psi}{\partial t}$. Another maximal acceleration called $\mathbf{u}_\Theta$ will induce $\frac{\partial \Theta}{\partial t}$ when $\mathbf{u}_\Theta$ is orthogonal to both $\mathbf{v}$ and $\mathbf{u}_\Psi$.



(a) The heading angle (Ψ) case

(b) The pitch angle (Θ) case

Figure 3-8: Velocity triangles used to determine when the linearized plant is no longer valid.

The velocity triangles in Figure 3-8 are used to calculate the minimum times in

which the angles $\Theta \pm \theta$ and $\Psi \pm \psi$ can be reached in the MILP problem:

$$T_\Theta = \frac{\|\mathbf{v}_\Theta\| \tan\Theta}{\|\mathbf{u}_\Theta\|} \tag{3.10}$$

$$T_\Psi = \frac{\|\mathbf{v}_\Psi\| \tan\psi}{\|\mathbf{u}_\Psi\|} \tag{3.11}$$

At acceleration $\mathbf{u}_\Theta$, the plant will no longer be valid in time T_Θ. At acceleration $\mathbf{u}_\Psi$, the plant will no longer be valid in time T_Ψ. Therefore, the plant can be applied to all point n where $T(n) < T_\Theta$ and $T(n) < T_\Psi$. The plant is discretized using a zero-order hold approximation as used in Section 2.3.1 and replaces the double integrator state equations where applicable.

3.6 Minimum Time

The minimum time MILP path planning problem is introduced in [37] for the fixed horizon case and used in [4] for receding horizon cases. Here we will slightly modify the formulation to work with a receding horizon. The vehicle, with position vector $\mathbf{x}$ has a destination $\mathbf{x}_{\text{goal}}$. A small box with half side-length δg is made around the goal point so that the MILP solution is not overly burdened by the goal constraint. If the vehicle is a rotorcraft and the goal state is a hover point, the δg can be arbitrarily small. If there are minimum speed constraints or if the goal state is serving as a fly through point, δg should be expanded according to the equations presented in Section 2.8.1. The constraints for this scenario are:

Figure 3-9: Expansion of the minimum time goal state shown in two dimensions.

$$\min \sum_{n=1}^{N} b_{\text{goal},i} W_i \tag{3.12}$$

subject to

$$\begin{aligned}
x_n - x_{\text{goal}} + \delta t &\leq M(1 - b_{\text{goal},i}) \\
x_n - x_{\text{goal}} - \delta t &\geq -M(1 - b_{\text{goal},i}) \\
y_n - y_{\text{goal}} + \delta t &\leq M(1 - b_{\text{goal},i}) \\
y_n - y_{\text{goal}} - \delta t &\geq -M(1 - b_{\text{goal},i}) \\
z_n - z_{\text{goal}} + \delta t &\leq M(1 - b_{\text{goal},i}) \\
z_n - z_{\text{goal}} - \delta t &\geq -M(1 - b_{\text{goal},i})
\end{aligned} \tag{3.13}$$

$$\sum_{n=1}^{N} b_{\text{goal},i} \leq 1 \tag{3.14}$$

The arrival constraint is relaxed at each time step i when $b_{\text{goal},i} = 0$. Because of the constraint on the summation of goal binaries, the arrival constraint can only be activated at one time step. Using a weighting scheme where $W_i > W_{i+1}$, the goal constraint will be activated for the first time step where the position vector lies within the goal box. Note that using this formulation, the path is not required to fly through the goal point at all. If the desire it to force the path to go through the goal point, then Equation 3.14 should be changed to an equality constraint. Be aware that this will result in problem infeasibilty when the goal state is not reachable from the initial position.

3.7 Mission Planning Waypoints

We will look at the case where the vehicle is directed to fly through a set of ordered waypoints (called the mission planning waypoints). It is assumed that these mission planning waypoints are spaced far enough apart that a successful trajectory is pos-

sible. We will again define a "current" and "next" waypoint, as pictured in Figure 2-18. Mission planning waypoint capture, and management of the "current" and "next" list is typically handled by the mission management computer. The following algorithm is used to decide how to handle the "current" and "next" waypoints, in order to produce smooth trajectories through the points:

Algorithm 1 Decision process for waypoint handling

```
for (Each time step n) do
  if ("current" ∈ planning horizon) then
    set "current" as the fly-through point (using min-time constraints)
    if ("current" ∈ last MILP solution (i.e. the MILP plan passed through "current")) then
      set "next" as the goto point (using two-norm approximation)
    end if
  else
    set "current" as the goto point
  end if
end for
```

3.8 Other Considerations

3.8.1 Variable Scaling

MILP solvers are much more efficient when the variables in the problem are all of a similar scale [49]. In some cases it is possible for the entire problem to become unstable if certain variables become too large (see Section 5.2.2). This issue needs to be kept in mind when using a MILP path planner for long distance and or high altitude flights. In these cases it is possible for the x, y, and z position variables to grow very large in magnitude. The easiest solution to this problem is to re-center the entire problem to the origin at every solve iteration. A position bias equal to the initial position vector is subtracted from all of the obstacles, goto points, radar sites, etc. so that they are in the proper relative relation to the vehicle in the MILP problem formulation. The MILP is solved normally, and the position bias is then added to the solution waypoints before sending the waypoint list is sent to the guidance controller.

3.8.2 Re-plan Rate

The rate at which the MILP problem is updated and solved depends largely on the solution time, the dynamics of the system, and the properties of the environment. Clearly, the lower bounds on the re-plan rate is equal to the computation time required to update and solve the MILP. As shown in this thesis, simple 3D problems can be solved in the sub 100 ms range, but often it is not worthwhile to replan at such a rapid rate. The mechanics of the waypoint follower required that the waypoints be as spaced out as possible for high speed flight, so the tests using a longer re-plan rate usually turned out more favorably than the test using a re-plan rate of 1 second. Other MILP implementations have used re-plan rate of up to once every 10 seconds [48]. Vehicle with slow dynamics, such as blimps are large airplanes, are unable to react quickly to new waypoint, so the speed of response needs to be taken into account when deciding on a re-plan rate.

If the environment can be considered static there is no need to re-plan very often. If there are no moving obstacles or pop-up threats, then the vehicle does not need to be concerned about possible collisions, because the environment is known *a priori*. In an environment where there are other aircraft in proximity, the replan rate should not be below the minimum time to collision.

3.9 Summary

Adding additional constraints to the core MILP formulation can make the path planner much more capable and robust. The use of linear interpolation points to minimize obstacle expansion is a key method, which is especially important when the timesteps are large. Solution paths not available using the traditional formulation become feasible with this technique. Adding linear interpolation points to the problem affects solve time much less than adding more time steps to the problem. The ability to use more complex plant models for the first few timesteps can also be very useful when the vehicle dynamics are not modeled well as a double integrator.

Chapter 4

Nap of the Earth Flight

4.1 Introduction

This section outlines an efficient way to represent terrain data, and describes several ways to manage terrain following in the objective function.

Nap of the Earth flight is a useful tool for helicopters and small fixed wing air vehicles. The ability to stay close to the surface of the earth allows for decreased radar detectability, a muddled sound signature, and protection from ground fire.

4.2 Ground Avoidance

Previous MILP formulations have modeled the ground as a set of normal obstacles, but doing so can add many unnecessary binaries to the problem. This section will demonstrate a new way of handling square-grid terrain data. The ground terrain is represented as a triangulated irregular network (TIN) [28]. Each square in the grid is split into two triangles, and the ground is representated in the MILP as a set of triangular shaped planes (see Figure 4-1). Using some preprocessing alogrithms, the terrain is smoothed so that the number of planes required is minimized.

Terrain data for the USA with a grid-size of 10 meters is freely available over the internet [50], and higher resolution data is available commercially. The smaller resolution maps can enable more elaborate terrain following, but consequently the

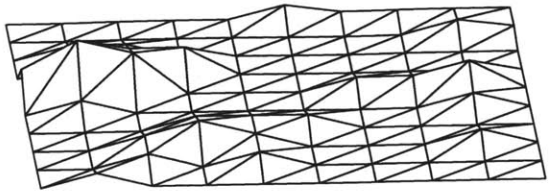


Figure 4-1: Terrain represented as a 10x10 grid of points connected by triangular planes.

MILP problem is also made more complex. We will first set a minimum size on the triangles. We want to ensure that the vehicle is not able to skip over an entire ground plane between MILP waypoints, because this could result in a dangerous situation. Thus, the vehicle cannot travel outside of the neighboring ground triangles (pictures in Figure 4-2) in one timestep. From Figure 4-2, we can see that the minimum distance to a non-neighboring ground plane is $a/\sqrt{2}$. So, the minimum ground triangle size over the entire simulation time is:

$$\max\left(|V|_{\text{sup}}(i)\,\Delta t(i)\right) \leq a/\sqrt{2} \tag{4.1}$$

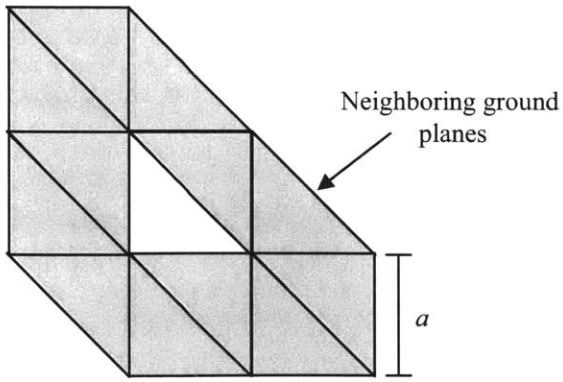


Figure 4-2: Each ground plane is surrounded by 12 neighboring ground planes.

4.2.1 Minimizing Triangle Count

When the terrain is largely unchaning over a swatch of land, it is not neccessary to model the terrain at the minimum resolution available. In order to minimize problem complexity, the triangles should be made as large as possible. Here we will present a straightforward algorithm that combines triangles in the TIN.

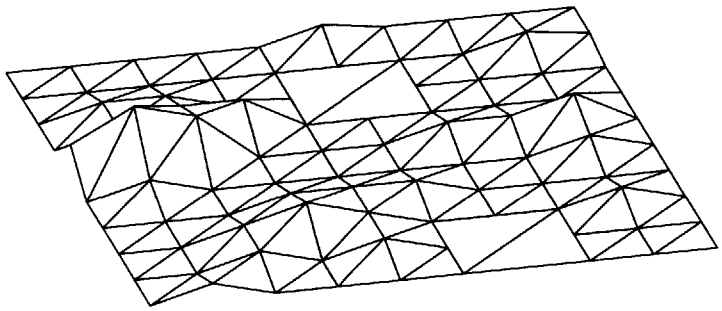


Figure 4-3: Some of the ground planes are combined to minimize complexity.

Algorithm 2 Minimizing planes in the terrain representation

Require: step size Δg

Require: $x = \{x_{min}, x_{min} + \Delta g, \ldots, x_{max}\}$, $y = \{y_{min}, y_{min} + \Delta g, \ldots, y_{max}\}$

Require: ground terrain map $z(x, y)$ defined at all points

```
for i = x do
  for j = y do
    if (All corners (z(i+1,j+1),z(i+1,j-1),z(i-1,j+1),z(i-1,j-1)) exist) then
      Define M as the set of all lines m_k intersecting through two neighboring
      points and the current point (i,j,z(i,j))
      if (m_k > z(i,j) then
        Remove z(i,j)
      end if
    end if
  end for
end for
```

The ground surface is discretized into a set of points. The points are then made into a set of planes, as defined by three points. Vehicle constaints in the MILP formulation as setup such that the vehicle must be above the plane for each timestep. Give three points, the plane is defined below.

$$\begin{vmatrix} x & y & z & 1 \\ x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \end{vmatrix} = 0 \tag{4.2}$$

4.2.2 Safety Considerations

Before the ground triangles are ready for addition into the MILP problem, some safety factors needed to be added in. The first step in handling the terrain data is to add a safety clearance that takes into account the size of the vehicle. This creates a configuration space, and is identical to the first expansion step used for regular obstacles.

The problem described in Section 2.4.1 regarding the vehicle flying though corners of obstacles must also be accounted for with the ground triangles. When the ground is locally concave-down, it possible for the vehicle to travel into unsafe space. Figure 4-4 demonstrates this. In this section we will present an algorithm to raise the terrain so that the vehicle will stay safe between MILP waypoints. First, we will define some terms. Each ground plane is surrounded by a number of *neighboring ground planes*, as pictured in Figure 4-2. A neighboring plane is any ground plane that has at least one point adjacent to the plane in question (which we will call the *current plane*). If atleast one neighboring plane forms a concave-down angle with the current plane, then a bias must be added to the current plane. The angle γ is the maximum angle between the current plane and all the neighboring planes that form a concave-down surface. Figure 4-5 shows the maximum incursion possible for some angle γ. Using this, we can see that each point defining the current plane needs to include a bias of:

$$d_{\text{incur}} = \frac{\|V\|_{n\ \max}\Delta t_n}{2\tan(\gamma/2)} \tag{4.3}$$

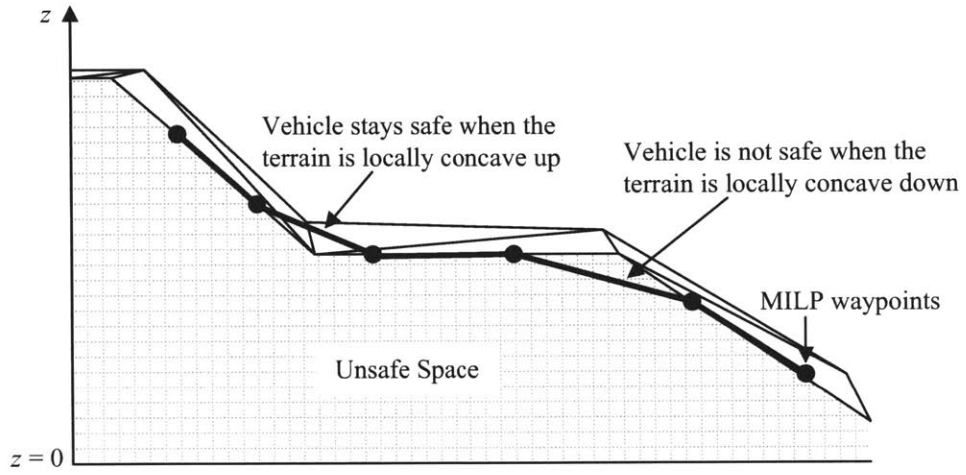


Figure 4-4: This two-dimensional cutaway of the terrain represenation shows why the terrain expansion is required.

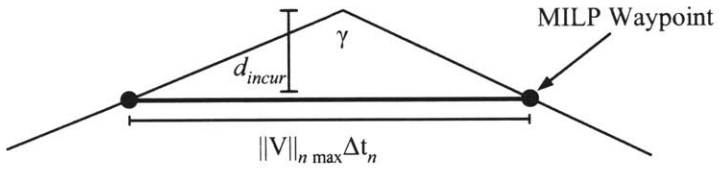


Figure 4-5: Two-dimensional cutaway of two ground planes showing the maximum ground incursion possible for a given angle γ

4.3 Altitude above ground level

One goal of nap of the Earth flight is to keep the trajectory closely matched to the local ground terrain. This intention needs to be added into the MILP's objective function, but simply seeking to minimize the altitude variables ($\min \sum_{n=1}^{N} z_n$) will not produce the intended result. Such an objective function will cause the vehicle to seek out lower terrain, not necessarily fly close to the local terrain. This issue is

resolved by instead defining an altitude above ground level (AGL) variable at each timestep and minimizing that instead.

$$\min \sum_{n=1}^{N} \mathrm{AGL}_n \tag{4.4}$$

The vehicle is above plane P at time n. The plane equation is $Ax + By + Cz = D$. Rearrage to the equation to get $z_{grnd} = D/C - A/Cx - B/Cy$. AGL_n is the z state variable minus $z_{grnd}(x, y)$.

$$\mathrm{AGL}_n = z_n - D/C + A/C\, x_n + B/C\, y_n \tag{4.5}$$

where A, B, C, and D are defined based on the current plane that the vehicle is over at timestep n.

4.4 Nap of the Earth Simulation

In this simulation, a large initial timestep of ten seconds was used. This large value was used not because of solve time, but to simulate a realistic MILP implementation on a medium to large UAV. Such vehicles have complex guidance and mission managment software, and sending waypoint at a high rate can induce large transients into the system. The vehicle started on one side of a ridgeline, and a goal point was provided on the other side of the ridge. Using a minimum distance-to-goal, minimum acceleration objective function, the vehicle reached the goal state in 90.6 seconds. The MILP re-planned 8 times with an average solve time of 1.235 seconds. The resulting trajectory is displayed in Figure 4-6.

4.5 Summary

By handling terrain data slightly different than regular obstacles, the number of binaries in the problem is reduced. By carefully sizing and raising the ground planes, it is possible to guarantee safety between MILP points in a nap of the Earth flight

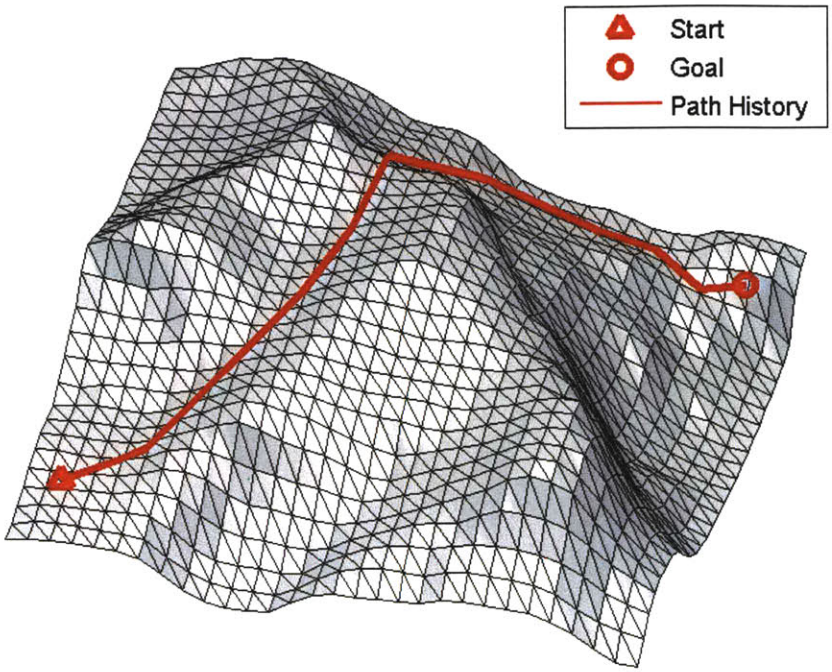


Figure 4-6: The air vehicle navigates over a ridgeline.

regime. This terrain formulation is used along side the horizon minimization technique to ensure that unneeded ground planes are not added to the problem formulation. A computer simulation shows that MILP can perform well for a long-range nap of the Earth mission.

Chapter 5

Hardware and Software Architecture

5.1 Introduction

This chapter covers some of the implementation details for both the computer simulations and the test flights described in this thesis. Most importantly, the "warm-start" concept for MILP path planning is introduced as a way to significantly decrease execution time.

5.2 Solver Software

5.2.1 ILOG CPLEX

The MILP solver used in all the computer simulations and test flight is CPLEX [21], a mathematical programming suite produced by ILOG. CPLEX is capable of solving linear programs, mixed integer linear programs, and even quadratic integer programs rapidly. Simple MILPs can be solved on the order of milliseconds. CPLEX's offer of fast computation time, robust error reporting, and effective memory management made it the preferred solver for the applications in this thesis. Open source solvers, such as lp_solve [9], are beginning to show promise, but their performance character-

istics still compare poorly to CPLEX.

5.2.2 Binary Variables

The current version of CPLEX (10.0) has a new feature that allows for what ILOG calls "indicator constraints." Indicator constraints match a binary variable with a linear equation, with the effect that the linear equation is inactive when the binary variable is switched off. This type of constaint gets rid of the need to use the "big M" formulation for the purpose of deactivating linear equations when certain conditions are (or are not) met. The big M method can lead to instabilities in some cases. For example, take $x1$, $x2$, and $x3$ as continuous variables and $b1$ as the binary variable used to inactive the problem using a big M formulation. For this example we will set $M = 10^9$. If then the left hand side of Equation 5.1 is close to 10^9, then the problem will become unstable, since the binary switch cannot be trusted as a way to deactivate the linear equation [22].

$$x1 + 3x2 + 2x3 \geq 5 + Mb1 \tag{5.1}$$

The indicator constraint removes the big M from the problem and simply enforces the rule that if the binary is switched on, the the associated linear equation is active. The potential for instability is removed, and solve time can also be decreased since using very large M values causes more computational complexity when relaxing constraints. The indicator constraint would be written as:

$$b1 \rightarrow 0; x1 + 3x2 + 2x3 \geq 5 \tag{5.2}$$

5.3 Code Implementation

Initial development of the MILP formulations described in this thesis were coded in Java. This choice is a departure from many of the previous implementation efforts, which often used a combination of MATLAB and AMPL [16, 30]. Java's object-

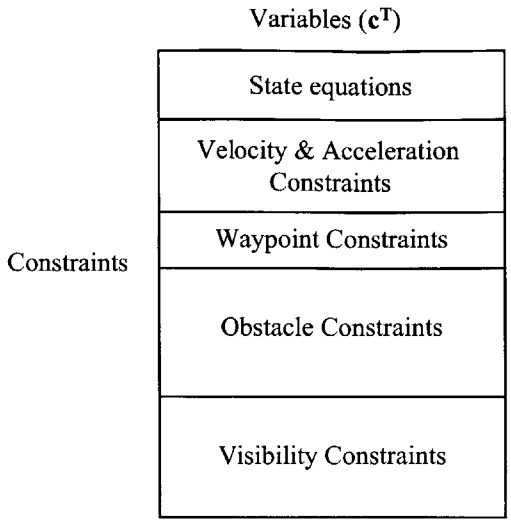


Figure 5-1: Representation of the constraints and variables in CPLEX. The entire rectange is the A matrix as defined in Equation 2.1.

oriented architecture simplified the process of adding and removing variables and constraints into the CPLEX environment. Use of the Java 3d API also allowed for real-time observation of simulation results, which is an ideal way to provide feedback on the quality of the MILP formulation.

Unfortunately, the memory overhead required by Java programs cause slower execution time, and there is seems to be more instability with a Java implementation. A C or C++ code base would probably be the best option for an onboard MILP path planner.

5.3.1 Warm-start

The MILP problem is formulated and solved in a multi-threaded environment. This structure enables CPLEX to be left open between each solve cycle. Many constraints and variables are constant in the MILP formulation, so it is most efficient to formulate the MILP, solve it, and then update CPLEX with a few new constraints and initial conditions before solving again. This warm-start approach increases overall solution speed because the cost of starting a new program, allocating memeory, opening CPLEX, and initializing a new CPLEX problem only occurs once during an entire

simulation. Note that previous papers on MILP path planning return the *solution time* for a MILP problem, which is a number that is usually tracked by the solver software. The warm-start approach will not change solution time, but it will decrease the *execution time* at each solution interation, which is a more accurate metric to track performance with. Execution time is the total amount of computation time required to solve the MILP problem. It includes initialization and update time, while solution time does not include these important quantities. Figure 5-2 shows how warm-start can decrease the execution time by over 0.3s at each iteration.

Algorithm 3 CPLEX interaction process using a warm-start

```
Initialize CPLEX environment
Create a new problem in CPLEX
Add all of the variables and constraints
Solve the CPLEX problem
while (!done) do
  if (Time to start solve process) then
    Get current vehicle state
    Update CPLEX problem with new initial conditions
    Calculate horizon size
    Add new obstacle and visibility constraints
    if (Objective changed) then
      Modify the problem's objective function
    end if
    Solve updated CPLEX problem
    Remove obstacle contraints
    Remove visibilty constraints
    Remove any other stale constraints
  end if
end while
Clear problem from CPLEX environment
Terminate the CPLEX
```

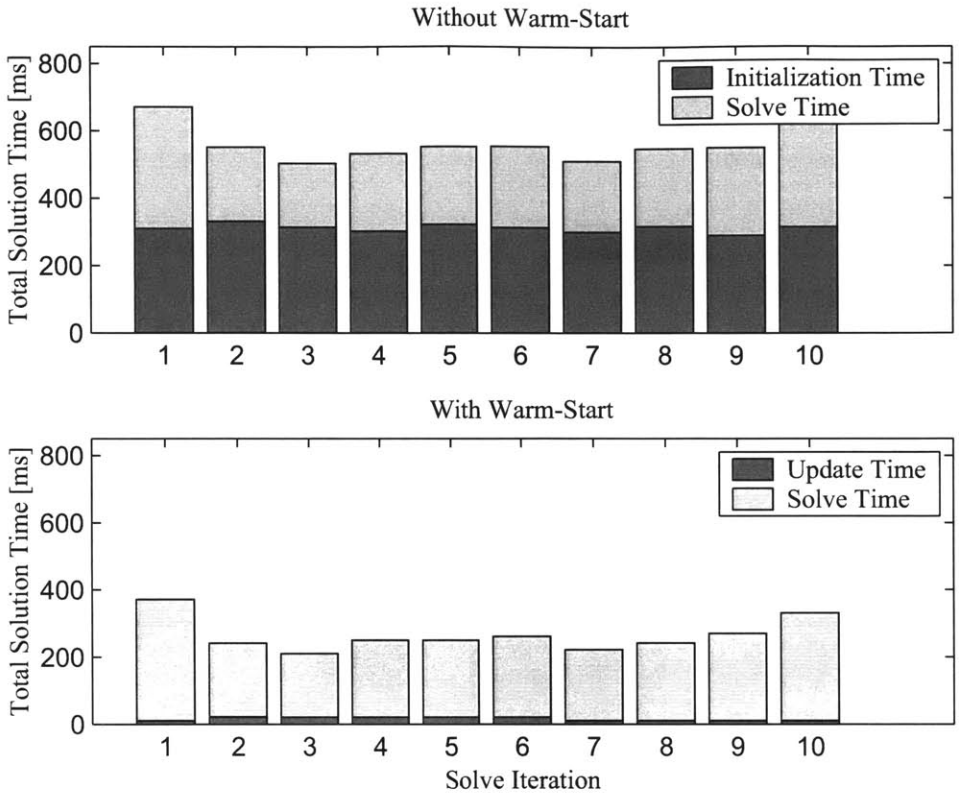


Figure 5-2: Using the warm start capability signficantly decreases solve time.

The cost of removing and adding particular variables and constraints from the CPLEX problem is much less computationally expensive than building up the whole problem again from scratch, as demonstrated in Figure 5-2.

5.4 Computer Hardware and Operating Systems

The MILP path planning software used in all test flight and computer simulations was run on Dell Latitude laptops. These machines were running Intel Pentium 4 Mobile CPU's with a processor speed of 2 GHz. 1 GB of ram was installed on each machine. One machine was running Windows XP Service Pack 4, and the other was running Red Hat Linux. For the sake of comparison, an indentical simulation was run on both the Windows and Linux computers repeatedly. The simulation was a point-to-point trajectory taking the vehicle through a tightly spaced group of obstacles. The flight

path through this obstacle field is shown in Figure 5-3. The flight path was essentially the same for each run on both the Linux and Windows simulations.

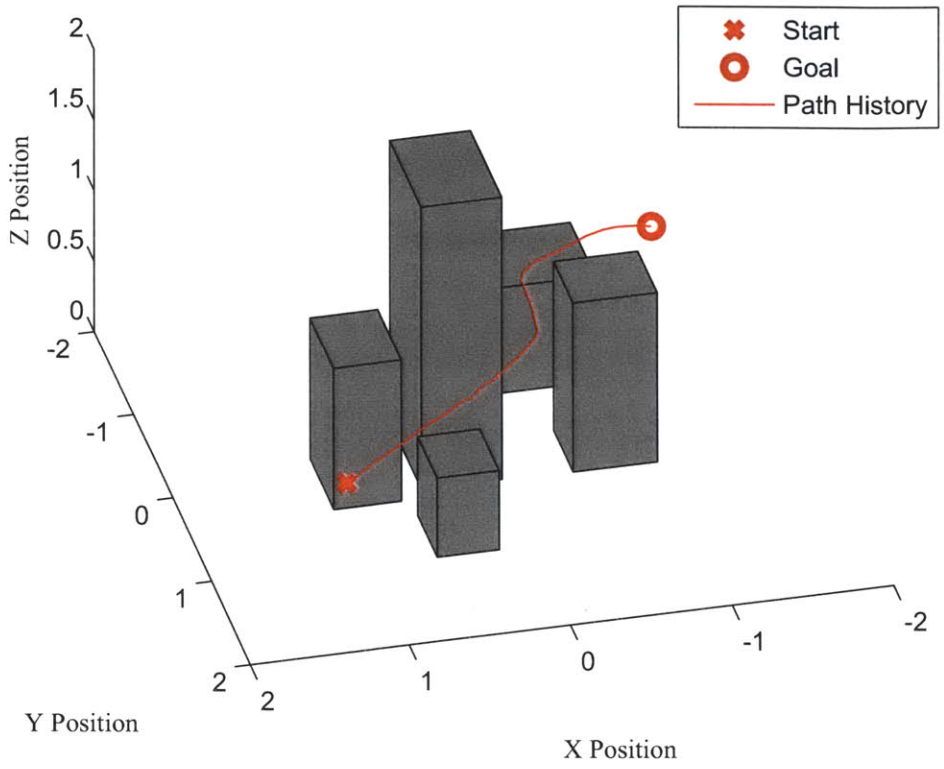


Figure 5-3: Simulation environment used for the sake of comparing solve times when using different computer operating systems with identical solver software.

All of the solution times are shown in Figure 5-4, and the result are averaged for each operating system in Figure 5-5. On average the MILP is solved 17.9% faster on the linux computer. Interestingly, the solution time for each solve iteration can very greatly between runs on the same operating system. This is most noticeable for the early solve iterations, where the solution time is longer. One reason for this effect has to do with the mechanics of the solver. The search through the binary tree is a quasi-random process, so completion time of the search for an optimal solution is also quasi-random, with respect to some upperbound on the solution time.

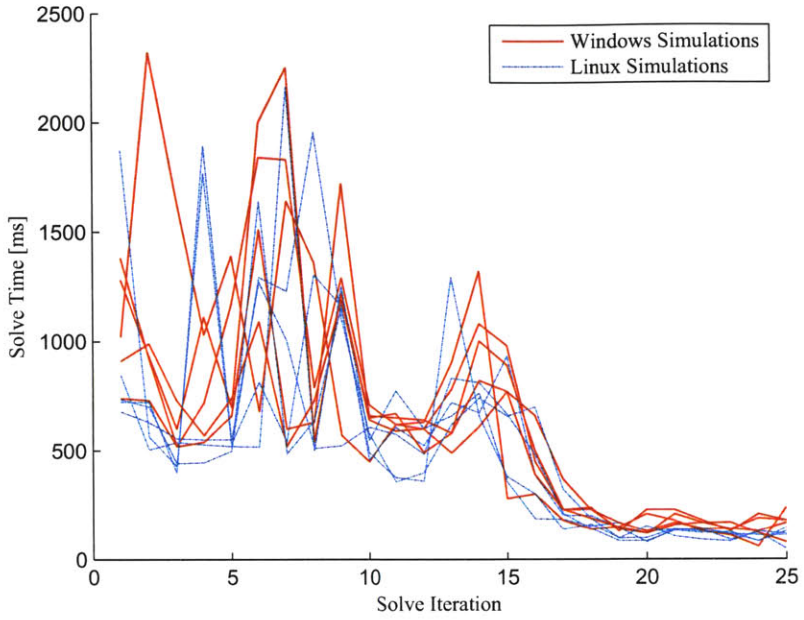


Figure 5-4: An identical simulation is run five times on both a Windows system and a Linux system.

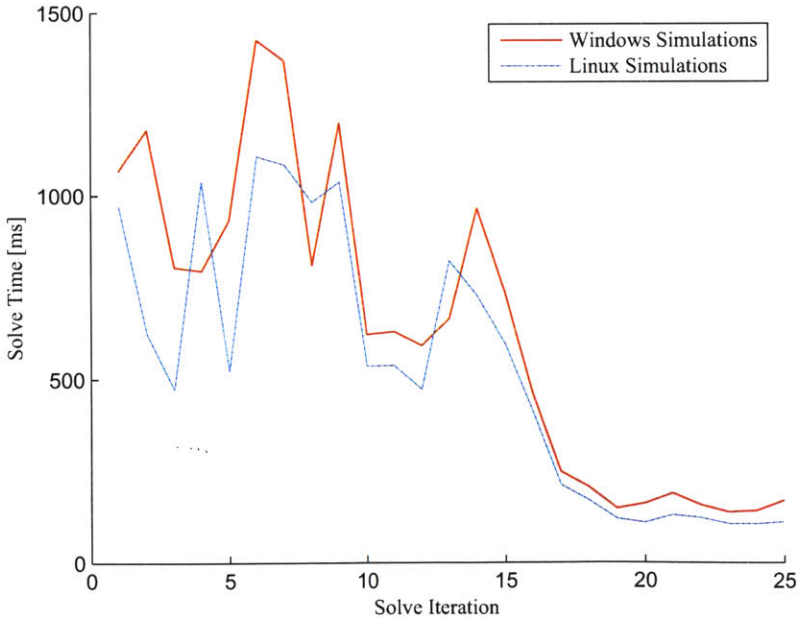


Figure 5-5: The mean solve time for each solve iteration is shown for the Windows and Linux cases.

5.5 Summary

The MILP path planner is more than just a collection of variables and constraints. The implementation details are critical, too. By selecting the right solver software, computer hardware, and coding language, the performance off a MILP path planner can be altered signficantly. Using a warm-start can be advantageous, especially in situation where a high re-solve rate is used.

Chapter 6

Test Flights & Simulations

6.1 Introduction

This chapter displays several test flights and computer simulations using the MILP capabilities described in the previous chapters. The first section focuses on flights with only one vehicle, and the second section goes on to show a multiple vehicle test. Problem size, solution time, and the resulting trajectories are shown. The space used for the quadrotor tests was fairly small, so long range trajectories or test with many obstacles were not feasible.

6.2 Vehicle Interface

Flights test were carried out on the indoor quadrotor platform at the Massachusetts Institute of Tecnology [52]. The quadrotor tests described in Chapter 6 used UDP message protocol to pass message between the computer solving the MILP problem and the quadrotor's GNC system. The information was sent as simple ASCII text. The UDP protocol was chosen over because of the increased speed. No packet drop-outs were apparent.

The MILP path planning computer received state information from the vehicle at a constant rate of once every 0.5 seconds. The state information included x,y and z postion values, v_x, v_y, and v_z velocity values, and a time stamp. The MILP computer

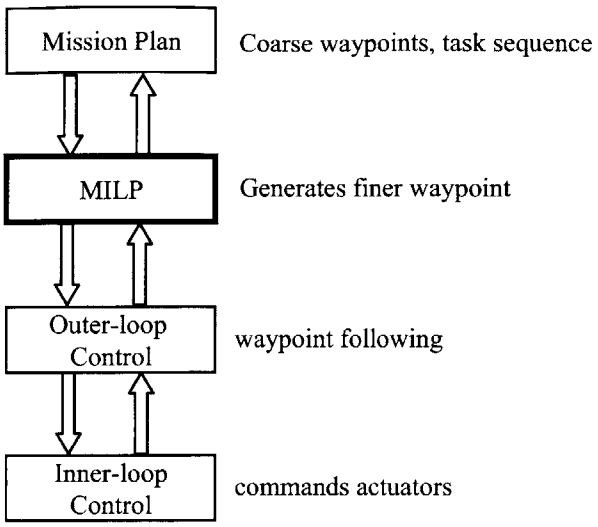


Figure 6-1: Different levels on guidance on an air vehicle

sends two waypoints after each solve interation, the "current" and "next" waypoints. Each of the waypoints contained the same variables state variables decribed above.

6.3 Single Vehicle Flights

6.3.1 Obstacle Avoidance

In this section, the quadrotor must avoid a single obstacle in the environment. The box is 0.8m high and 0.6m wide in both the x and y directions. The vehicle starts on one side of the box, and a goto point is defined on the other side of the box. The primary objective is for the vehicle to minimize distance to the goal, and also minimize acceleration, to a lesser extent. After running the simulations and test flights, it was clear that the vehicle would start heading towards the goal quickly, but would slow down as it got closer to the goal point. This is because as the distance to the goal becomes smaller, the relative objective on minizing the distance shrinks in relation to the minimize acceleration objective. To combat this effect, a minimum time constraint was added to the problem. The difference in arrival time is listed in Table 6.3.1. The minimum time test is shown in Figure 6-2. The status of the

simulation is shown as it is about half way to the goal, and the entire path history of the quadrotor flight path is also dispayed. Notice that the actual trajectory seems to fly outside of the simulated trajectory. This is due to the fact that the quadrotor does not respond to the waypoint commands as quickly as the simulation expects. The vehicle also overshoots the goal state because it is descending at a high rate of speed, and it is unable to accelerate upward fast enough when the goal state is reached.

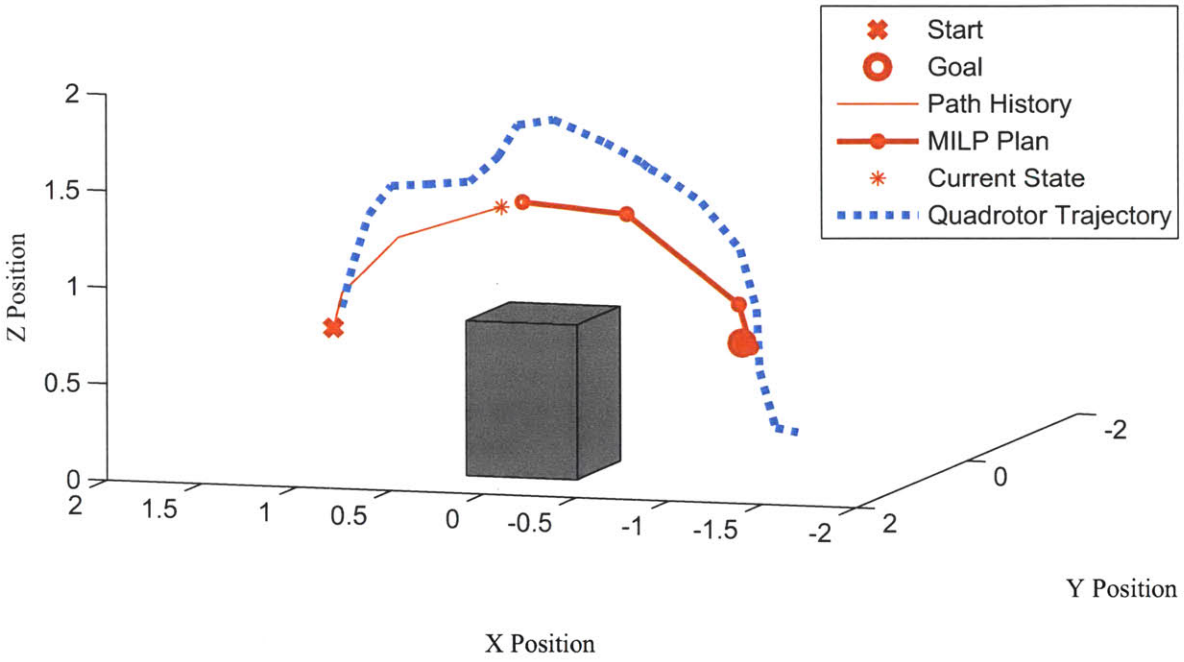


Figure 6-2: Computer simulation of a simple obstacle avoidance trajectory.

Table 6.1: MILP problem characteristics for the single vehicle obstacle avoidance scenario.

Test Type	Time To Goal	Mean Solve Time	Mean # of Variables	Mean # of Constraints
Simulation	$8.67s$	$91.71ms$	201	638
Simulation with min time	$7.24s$	$116.83ms$	207	675
Test flight	$11.63s$	$75.43ms$	162	592
Test flight with min time	$10.98s$	$69.4ms$	168	629

With altitude penalty

A penalty on altitude in included in the simulation. Whenever the altitude of the vehicle goes above 0.7m in the MILP solution, the objective cost rises signficantly. Notice in Figure 6-3 that now the vehicle flys around the obstacle instead of over it. Again, the actual path flies outside of the simulated path. The vehicle reaches the speeds it is commanded to, but is unable to track the turning commands precisely. With lower speeds tests, the trajectory following is much more precise.

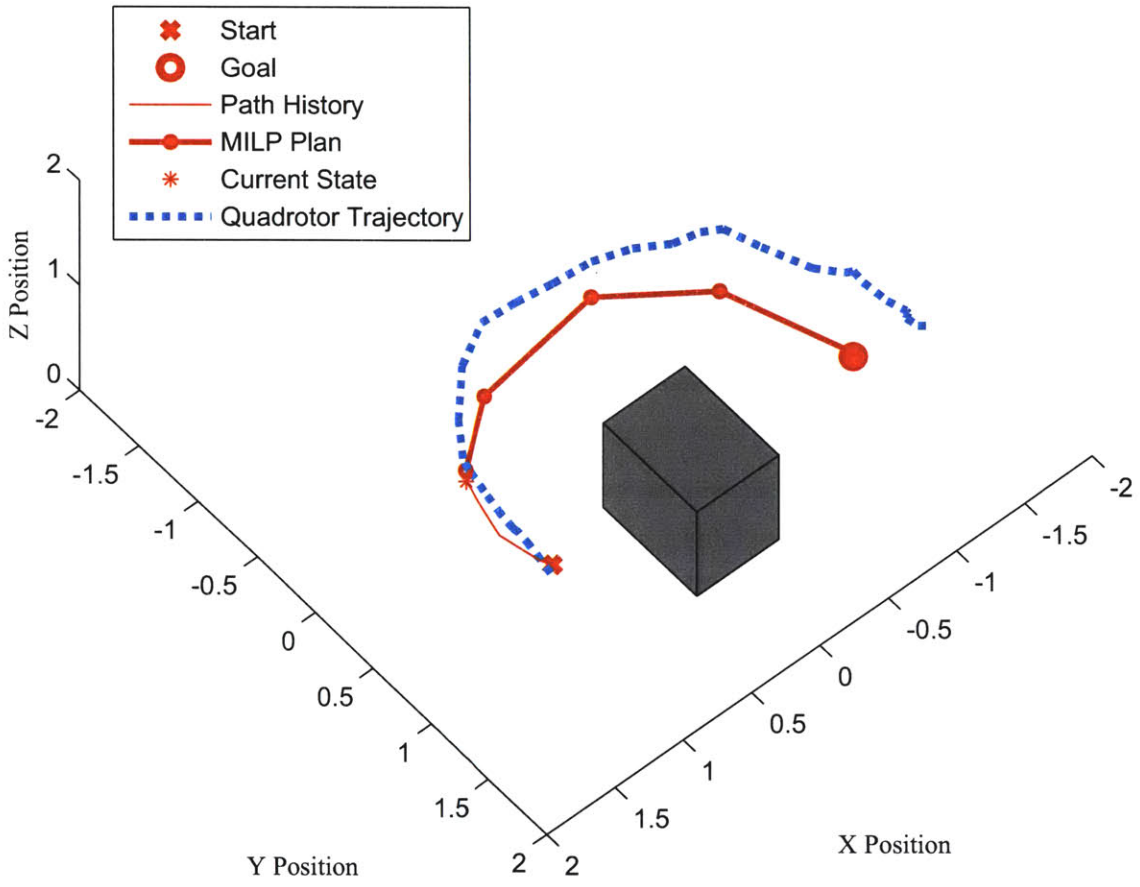


Figure 6-3: Computer simulation of the minimum time, altitude penalized scenario. Now the vehicles chooses to fly around the obstacle.

Table 6.2: MILP problem characteristics for the single vehicle, altitude penalized scenario.

Test Type	Time To Goal	Mean Solve Time	Mean # of Variables	Mean # of Constraints
Simulation	11.39s	152.66ms	206	642
Simulation with min time	10.00s	227.87ms	213	681
Test flight	15.31s	102.46ms	145	576
Test flight with min time	14.21s	105.42ms	152	609

6.4 Path Planning with Multiple Vehicles

The multiple vehicle scenario was run using a separate computer controlling each each vehicle over the network. Two vehicles were in the environment. Each was given a goal point, which were picked so that the vehicle would be forced to avoid each other while trying to reach their respective goal states. Each vehicle has limited knowledge about the other quadrotor. The MILP path planner receives an update of it's own state and the other vehicle's state every 0.5 seconds. The flight test was run at a slow speeds for safety reasons. As the vehicles passed each other, they were very close but never collided, thanks to the moving obstacle constraints. Figure 6-4 shows the flight path of each vehicle. The tracks appear slightly erratic. This is largely due to the slow speed and also the wind forces from the neighboring vehicle. The mean solve time on the Windows laptop for this flight test was 135ms and the maximum solve time of 400ms occured at the first solve iteration. The mean solve time on the Wlinux laptop was 113ms and the maximum solve time of 398ms also occured at the first solve iteration.

6.5 Bounds on Position Variables

In certain flight test or simulation environments, it is necessary to put constraints on the position variables in order that the vehicle stay within a defined region of space. The indoor quadrotor test flights required a bound on position variables so that the vehicles would not hit walls or fly outside of the camera system's view. One straightforward is option is to define the x, y, and z position variables as being bound

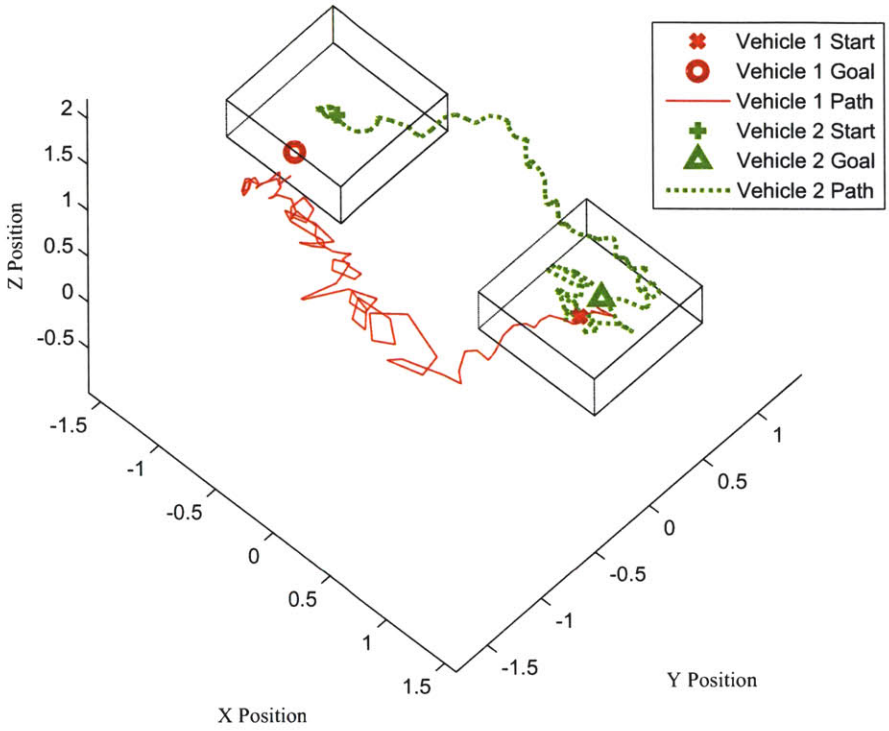


Figure 6-4: Results from a low-speed multi-vehicle de-confliction test. The dashed boxes show the size of the vehicles.

on a certain interval for all time steps:

$$\text{for } n = 1 \ldots N \tag{6.1}$$

$$\begin{aligned} x_n &\in [x_{\min}, x_{\max}] \\ y_n &\in [y_{\min}, y_{\max}] \\ z_n &\in [z_{\min}, z_{\max}] \end{aligned} \tag{6.2}$$

In simulations, this constraint worked without a problem, but during test flights a serious shortcoming became apparent in the formulation. If the vehicle strays too far from the MILP waypoints and goes outside of the position bounds, then it is possible that the MILP path planner will not be able to get the vehicle back into the safe region in one time step, resulting in an infeasible solution. The solution here is to

reform the position bounds as a soft constraint. The x, y, and z position variables are left free and a set of cost region constraints are used to keep the vehicle in a defined space:

$$\text{min: } \sum_{n=1}^{N} M\, b_n$$

subject to:

$$\begin{aligned}
x_n &\geq x_{\min} - M\, b_n \\
x_n &\leq x_{\max} + M\, b_n \\
y_n &\geq y_{\min} - M\, b_n \quad \text{for } n = 1 \ldots N \\
y_n &\leq y_{\max} + M\, b_n \\
z_n &\geq z_{\min} - M\, b_n \\
z_n &\leq z_{\max} + M\, b_n
\end{aligned} \tag{6.3}$$

$$\begin{aligned}
x_n &\in (-\infty, \infty) \\
y_n &\in (-\infty, \infty) \\
z_n &\in (-\infty, \infty)
\end{aligned} \tag{6.4}$$

Using the constraints above, the chance of producing an infeasible solution reduced. If the vehicle has no choice but to be outside the bounds, then it can do so, albeit with high cost. Since the set of constraints is convex, only one binary variable is needed at each time step. The penalty on solution time when adding these constraints is minimal, only a change of a few milliseconds was noticed.

6.6 Summary

Using an indoor test platform, the MILP path planner was tested for several different scenarios. Experimentally it was shown that MILP can successfully navigate an air vehicle around obstacle at a high rate of speed safely. MILP was also used to maintain separation between two vehicles using moving obstacle constraints.

Chapter 7

Conclusion

7.1 Summary

Using the *variable MILP* paradigm, many of the typical MILP path planner drawbacks, such as solve time, overly conservative trajectories, and innacurate plant representation, have been avoided or minimized. Using horizon minimization, unreachable objects are kept out of the problem formulation, thereby reducing the number of constraints and variables in the problem at each solve iteration. Variable timesteps are used to generate finely spaced waypoints in the near future and coarser waypoints further out in the path plan. This increases the length of the total simulation time without increasing the number of decision variables. Linear interpolation points reduces the need for extreme obstacle expansion when timesteps are spaced far apart, preventing trajectories from becoming too conservative. Special contraints for moving obstacles are used to perform vehicle de-confliction tasks. With these improvements, a MILP path planner can be used robustly on an air vehicle to navigate through an obstacle field or maneuver between a coarser set of waypoints.

The warm-start capability decreases execution time signficantly at each solve iteration, allowing fairly simple a MILP path planner to resolve the problem at an interval under one second with plenty of margin on computation time. Test flights on an indoor quadrotor testbed show that smooth, high speed trajectories can be produced with the MILP path planner running at a solve rate of under 1 second for

both obstacle avoidance and vehicle de-confliction scenarios.

7.2 Future Work

The MILP is an extremely efficient way to approach the path planning problem. One major drawback is that the resolution and features of the problem must be scaled down such that the MILP solve time is less than the replan rate. As computer speeds increase and solution algorithms improve, solve time will decrease and so the MILP path planner can be made more complex. For example, the two-norm approximation could be discretized to a higher level at each timestep. In this thesis the circle was never discretized into more than 8 pieces because anything more would result in a noticeable increase in execution time.

In instances where the dynamics or guidance system of a vehicle neccesitates that waypoint are fed at a slow rate (nominally a rate greater than 10 seconds), it may be reasonable to solve a second MILP problem in the background at a high rate. This higher rate MILP will check for potentially danger situations that could occure between the 10 second planning intervals, such as a popup threat. The higher rate MILP could send an emergency maneuver command to the control system if the vehicle is in danger.

7.2.1 Pointing Constraints

Low observable (LO or "stealth") aircraft are least observable to a radar when they are pointed straight at the radar emmitter. Recconnaissance aircraft with fixed sensors must keep the observation point at a specific heading with respect to the vehicle for the sake of collecting data. Using MILP for low observable path planning has been covered by [12]; but the solution times were far to slow to allow for real-time implementation.

As the speed at which complex MILP problems can be solved improves, it will be important to revisit this topic and determine if the capability for on-line LO path planning is a feasible task.

7.2.2 Beyond MILP

As solution algorithms improve, it may become more sensible to use quadratic integer programming instead of a MILP [32]. The movement from a linear to a quadratic framework makes the path planner much more capable. One advantage is that the quadratic terms make it much easier to use magnitudes and distances in the problem formulation. The most general form for a quadratically constrained program (QCP) is:

$$\begin{aligned} \text{Minimize } & \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + c^T\mathbf{x} \\ & \text{Subject to} \\ A\mathbf{x} &= b \\ a_i^T\mathbf{x} + \mathbf{x}^T Q_i\mathbf{x} &\leq r_i \text{for } i = 1, ..., q \\ l \leq \mathbf{x} &\leq u \end{aligned} \tag{7.1}$$

7.2.3 Stochastic Considerations

In the real world, information about the environment is never known precisely. Sensors only predict the presence of obstacles, terrain data could be inaccurate, and radar sites could be located in a different location that expected. It is possible to include these uncertainties and predictions into the MILP framework with some effort, and this could prove exceptionally useful for future air vehicles using MILP path planners in operational environments.

Bibliography

[1] E. Balas. Disjunctive programming. *Discrete Optimizations II, Annals of Discrete Mathematics*, 5, 1979.

[2] A. Bayen, C. Tomlin, Todd Callantine, Y. Ye, and J. Zhang. Optimal arrival traffic spacing via dynamic programming. *AIAA Conference on Guidance, Navigation, and Control*, August 2004.

[3] A. M. Bayern and Claire J. Tomlin. Real-time discrete control law synthesis for hybrid systems using milp: Application to congested airspace. *IEEE American Control Conference*, June 2003.

[4] J. Bellingham, A. Richards, and J.P. How. Receding horizon control of autonomous aerial vehicles. *Proceedings of the American Control Conference*, pages 3741–3746, May 2002.

[5] J. Bellingham, M. Tillerson, A. Richards, and J. P. How. Multi-task allocation and trajectory design for cooperating uavs. *Cooperative Control: Models, Applications and Algorithms at the Conference on Coordination, Control and Optimization*, pages 1–19, November 2001.

[6] R. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, NJ, 1957.

[7] R. Bellman. *Adaptive Control Processes.* Princeton University Press, Princeton, NJ, 1961.

[8] A. Bemporad and M. Morari. Robust Model Predictive Control: A Survey. *Robustness in Identification and Control*, 245:207–226, 1999.

[9] Michel Berkelaar. lp_solve, 2004. http://lpsolve.sourceforge.net/5.5/.

[10] J. T. Betts. Survey of numerical methods for trajectory optimization. *AIAA Journal of Guidance, Navigation, and Control*, 21(2):193–207, 1998.

[11] Carl B. Boyer and Uta C. Merzbach. *A History of Mathematics*, chapter The Age of Euler. Wiley, New York, NY, USA, second edition, 1991.

[12] A. Chaudhry, K. Misovec, and R. D'Andrea. Low observability path planning for an unmanned air vehicle using mixed integer linear programming. *IEEE Conference on Decision and Control*, December 2004.

[13] R. Diestel. *Graph Theory.* Springer, Berlin, DE, second edition, 2000.

[14] M G. Earl and R. D'Andrea. Iterative milp methods for vehicle control problems. *IEEE Transactions on Robotics*, 21(6), December 2005.

[15] C. A. Floudas. *Nonlinear and Mixed-Integer Optimization.* Oxford University Press, New York, NY, USA, 1995.

[16] R. Fourer, D. M. Gay, and B. W. Kernighan. *AMPL: a modeling language for mathematical programming.* Boyd and Fraser, Danvers, MA, USA, 1999.

[17] C. E. Garcia, D. M. Prett, and M. Morari. Model predictive control: theory and practice a survey. *Automatica*, 25(3):335–348, 1989.

[18] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, CA, USA, 1979.

[19] J. R. Guardiano. Pentagon pushing uav development. *Helicopter News*, 27(45), Decemember 2001.

[20] Yong K. Hwang and Narendra Ahuja. Gross motion planning a survey. *ACM Computing Survey*, 24(3):219–291, 1992.

[21] ILOG. Cplex, 2006. http://www.ilog.com/products/cplex/.

[22] ILOG. *ILOG CPLEX Users Guide*. 2006.

[23] D.S. Johnson and L.A. McGeoch. *Local Search in Combinatorial Optimization*, chapter The Traveling Salesman Problem: A Case Study in Local Optimization, pages 215–310. E.H.L. Aarts and J.K Lenstra (eds.), John Wiley & Sons, New York, NY, USA, 1997.

[24] E. King, M. Alighanbari, and J. P. How. Experimental demostration of cooridinated control for multi-vehicle teams. *Proceedings of the 16th IFAC Symposium on Automatic Control in Aerospace*, June 2004.

[25] Y. Kuwata. Real-time trajectory design for unmanned aerial vehicles using receding horizon control. Master's thesis, Massachusetts Insitute of Technology, Department of Aeronautics and Astronautics, June 2003.

[26] Y. Kuwata and J. P. How. Three dimensional receding horizon control for uavs. *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, August 2004.

[27] Y. Kuwata, A. Richards, T. Schouwenaars, and J. P. How. Decentralized robust receding horizon control for multi-vehicle guidance. *Proceedings of the American Control Conference*, pages 2047–2052, June 2006.

[28] C. Ma and R. Miller. Milp optimal path planning for real-time applications. *Proceeding of the 2006 American Control Conference*, 2006.

[29] S.M. Malaek and A. Abbasi. Near-optimal terrain collision avoidance trajectories using elevation maps. *IEEE Aerospace Conference*, March 2006.

[30] The Mathworks. Matlab, 2006. http://www.mathworks.com/products/matlab/.

[31] L. R. Newcome. *Unmanned Aviation: A Brief History of Unmanned Aerial Vehicles*. American Institute of Aeronautics and Astronautics, Reston, VA, USA, 2004.

[32] M. E. O'Kelley. A quadratic integer program for the location of interacting hub facilities. *European Journal of Operational Research*, 32(3):393–404, 1987.

[33] L. Pearson. *Naval Aviation in World War I*, chapter Developing the Flying Bomb, pages 70–74. Chief of Naval Operations, Washington, D.C., USA, 1969.

[34] S. J. Qin and T. A. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7):733–764, July 2003.

[35] S. Rathinam, R. Sengupta, and S. Darbha. A resource allocation algorithm for multi-vehicle sytems with non-holonomic constraints. *Institute of Transportation Studies Research Reports*, may 2005.

[36] J. H. Reif. Complexity of the movers problem and generalizations. *Proceedings of the 20th IEEE Symposium on the Foundations of Computer Science*, pages 421–427, 1979.

[37] A. Richards and J. How. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. *Proceedings of the IEEE American Control Conference*, pages 1936–1941, May 2002.

[38] A. Richards and J. P. How. Decentralized model predictive control of cooperating uavs. *Proceedings of the IEEE Conference on Decision and Control*, pages 4286–4291, 2004.

[39] A. Richards and J. P. How. Mixed-integer programming for control. *Proceedings of the 2005 American Control Conference*, pages 2676–2683, June 2005.

[40] A. Richards, Y. Kuwata, and J.P. How. Experimental demonstrations of real-time milp control. *AIAA Guidance, Navigation, and Control Conference and Exhibit*, August 2003.

[41] A. Richards, T. Schouwenaars, J. P. How, and E Feron. Co-ordination and control of multiple uavs. *AIAA Conference on Guidance, Navigation, and Control*, August 2002.

[42] T. Schouwenaars, , M. Valenti, E. Feron, and J. P. How. Implementation and flight test results of milp-based uav guidance. *Proceedings of the IEEE Aerospace Conference*, February 2005.

[43] T. Schouwenaars. Mixed integer programming for optimal collision-free path planning of autonomous vehicles. Master's thesis, Katholieke Universiteit Leuven, Department of Electrical Engineering, May 2001.

[44] T. Schouwenaars. *Safe Trajectory Planning of Autonomous Vehicles.* PhD dissertation, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, February 2006.

[45] T. Schouwenaars, B. DeMoor, E. Feron, and J. How. Mixed integer programming for multi-vehicle path planning, 2001.

[46] T. Schouwenaars, J. P. How, and E. Feron. Decentralized cooperative trajectory planning of multiple aircraft with hard safety guarantees. *Proceedings of the AIAA Guidance, Navigation and Control Conference*, August 2004.

[47] T. Schouwenaars, J. P. How, and E. Feron. Receding horizon path planning with implicit safety guarantees. *Proceedings of the 2004 American Control Conference*, July 2004.

[48] T. Schouwenaars, J. P. How, and E. Feron. Multi-vehicle path planning for non-line of sights communications. *IEEE American Control Conference*, June 2006.

[49] Alexander Schrijver. *Theory of linear and integer programming.* John Wiley & Sons, Inc., New York, NY, USA, 1986.

[50] United States Geological Survey. Digital elevation model data, 2006. http://data.geocomm.com/dem/.

[51] Paolo Toth and Daniele Vigo. *The vehicle routing problem.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

[52] M. Valenti, B. Bethke, G. Fiore, J. P. How, and E. Feron. Indoor multi-vehicle flight testbed for fault detection, isolation, and recovery. *AIAA Guidance, Navigation, and Control Conference*, 2006.

[53] M. Valenti, T. Schouwenaars, Y. Kuwata, E. Feron, J. P. How, and J. Paunicka. Implementation of a manned vehicle-uav misison system. *In Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, August 2004.

[54] R. J. Vanderbei. *Linear Programming Foundations and Extensions.* Kluwer Academic Publishers, 1997.

[55] L. Vlacic, M. Parent, and F. Harashima. *Intelligent Vehicle Technologies: Theory and Applications.* Society of Automotive Engineers, Warrendale, PA, USA, 2001.

[56] H. P. Williams and S.C. Brailsford. *Advances in Linear and Integer Programming*, chapter Computational logic and integer programming, pages 249–281. Oxford Science Publications, Oxford, 1996.

[57] Bill Yenne. *Attack of the Drones: A History of Unmanned Aerial Combat.* Zenith Press, St. Paul, MN, USA, 2004.

