Robust Trajectory Planning for Unmanned Aerial Vehicles in Uncertain Environments

by

Brandon Luders

B.S., Aerospace Engineering
Georgia Institute of Technology (2006)

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

September 2008

Author .....
Department of Aeronautics and Astronautics
August 28, 2008

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

Accepted by .....
Prof. David L. Darmofal
Associate Department Head
Chair, Committee on Graduate Students

Robust Trajectory Planning for Unmanned Aerial Vehicles in Uncertain Environments

by

Brandon Luders

Submitted to the Department of Aeronautics and Astronautics
on August 28, 2008, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

As unmanned aerial vehicles (UAVs) take on more prominent roles in aerial missions, it becomes necessary to increase the level of autonomy available to them within the mission planner. In order to complete realistic mission scenarios, the UAV must be capable of operating within a complex environment, which may include obstacles and other no-fly zones. Additionally, the UAV must be able to overcome environmental uncertainties such as modeling errors, external disturbances, and an incomplete situational awareness. By utilizing planners which can autonomously navigate within such environments, the cost-effectiveness of UAV missions can be dramatically improved.

This thesis develops a UAV trajectory planner to efficiently identify and execute trajectories which are robust to a complex, uncertain environment. This planner, named Efficient RSBK, integrates previous mixed-integer linear programming (MILP) path planning algorithms with several implementation innovations to achieve provably robust on-line trajectory optimization. Using the proposed innovations, the planner is able to design intelligent long-term plans using a minimal number of decision variables. The effectiveness of this planner is demonstrated with both simulation results and flight experiments on a quadrotor testbed.

Two major components of the Efficient RSBK framework are the robust model predictive control (RMPC) scheme and the low-level planner. This thesis develops a generalized framework to investigate RMPC affine feedback policies on the disturbance, identify relative strengths and weaknesses, and assess suitability for the UAV trajectory planning problem. A simple example demonstrates that even with a conventional problem setup, the closed-loop performance may not always improve with additional decision variables, despite the resulting increase in computational complexity. A compatible low-level troller is also introduced which significantly improves trajectory-following accuracy, as demonstrated by additional flight experiments.

Thesis Supervisor: Jonathan How
Title: Professor

Acknowledgments

First, I would like to thank Professor How for being a constant source of guidance and wisdom in my research pursuits at MIT, leading to this thesis. Regardless of the problem I encounter in my work, he always several handfuls of ideas to try. Professor How has played a critical role in helping me identify my personal research interests and apply them to projects such as this thesis.

This research has been funded by the Air Force Office of Scientific Research, USAF, under grant FA9550-08-1-0086.

Much of the work related to robust model predictive control (RMPC) in this thesis was inspired by the prior work and discussions of Professor Arthur Richards and Dr. Yoshiaki Kuwata, both of whom have provided valuable insights. Through frequent report revisions and e-mail exchanges, they (along with Professor How) have been essential in refining the state of our current RMPC research.

The members of the Aerospace Controls Laboratory have been extremely supportive throughout my work on this thesis, whether through proofreading this document, helping me reason through a problem, or simply offering helpful advice. I would like to offer a special thanks to Brett Bethke and Josh Redding, who graciously offered to read this document and suggest revisions. I would also like to thank Spencer Ahrens, Luc Brunet, Cameron Fraser, and Frant Sobolic for the specific ways in which they have assisted me. Above all, I would like to thank Kathryn Fischer for her support and resourcefulness throughout my time here.

Finally, my family has been a constant source of love and inspiration for me throughout my education. So, to my parents, Brian and Lori, and my brother Chris, thank you for always being there for me and supporting my aspirations.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The unmanned aerial vehicle (UAV) has taken a more prominent role in aerial missions over the last decade, as emerging technology has enabled its successful operation in more complex scenarios. The primary advantage of the UAV is the absence of a human occupant, often resulting in simpler vehicle designs and less expensive production compared to manned vehicles. Furthermore, UAVs can be used to perform missions which may be too dangerous for a human presence, such as military operations in hostile territory or long-duration reconnaissance. In addition to these scenarios, UAVs may also be used for search-and-rescue, traffic and weather monitoring, and urban surveillance.

Even as new capabilities are developed, a diverse collection of UAVs has already been deployed for a variety of mission scenarios. Some of the current bounds of UAV operation are demonstrated by the vehicles shown in Figs. 1-1 to 1-4. The Northrop Grumman RQ-4 Block 10 Global Hawk (Fig. 1-1) is one of the largest UAVs currently in deployment, with a mass of 12,000 kg and a 35-m wingspan. With a range of over 20,000 kilometers, the Global Hawk UAV can provide real-time imaging for broad mission-level intelligence, surveillance, and reconnaissance (ISR) [2]. The General Atomics Aeronautics Systems Mariner (Fig. 1-2), used for persistent ISR in maritime environments, can perform missions exceeding 45 hours in duration [3]. The Composite Engineering AQM-37 (Fig. 1-3), used to simulate supersonic ballistic missile threats, has a ceiling of over 35 kilometers [4].

Figure 1-1: Northrop Grumman RQ-4 Block 10 Global Hawk

Figure 1-2: General Atomics Aeronautics Systems Mariner

Figure 1-3: Composite Engineering AQM-37

Figure 1-4: AeroVironment WASP II

At the other extreme is the development of "micro" air vehicles (MAVs), capable of providing squad-level support in local and possibly cluttered environments. One such vehicle already in operation is the 290-gram AeroVironment WASP II (Fig. 1-4). Though hand-launched, the WASP II MAV can use data from the Global Positioning System (GPS) to navigate autonomously within a 2-kilometer radius. Each of these UAVs has found a useful role in aerial missions, but their impact remains limited by the amount of autonomy which can be achieved with current capabilities.

1.1 Motivation

In order to complete realistic mission scenarios, the UAV must be capable of operating within complex and/or uncertain environments. The environment may include several types of no-fly zones, associated with physical obstacles, radar sites, and other threats to the vehicle. The predicted vehicle model, typically used to develop a control

strategy, may be subject to internal uncertainties (such as modeling errors) and/or external uncertainties (such as wind) which cause the vehicle to deviate from its expected trajectory. The vehicle's knowledge of its environmental constraints may be incomplete and/or uncertain, depending on the vehicle's ability to perceive its surroundings. Because the UAV's situational awareness may change over time, its ideal trajectory may change, as well. For example, the UAV may identify and begin to execute a specific trajectory, only to later discover an obstacle which renders that trajectory infeasible.

Given this environmental uncertainty, it is often necessary to identify UAV trajectories through on-line plan generation, rather than an off-line pre-calculation. For this reason, most UAV missions currently require extensive human support to monitor UAV subsystems, track mission progress, react to environmental changes, and possibly fly the vehicle remotely. However, with typical operator-to-vehicle ratios of at least 2:1 [5], this paradigm limits the cost-effectiveness of UAV missions. Though the threat to human operators is removed, a UAV mission often requires significantly expanded operations compared to an equivalent mission with a manned vehicle.

A critical need in UAV research, then, is to improve mission cost-effectiveness by identifying ways to increase the level of autonomy. With appropriate health management tools, a UAV could analyze its own subsystem data and respond appropriately to off-nominal conditions or system failure. A team of UAVs could make decisions on how to complete a task list through collaborative, decentralized assignment algorithms. Similarly, each UAV, given a task, should be able to use its own situational awareness to design efficient, robust trajectories to mission waypoints. By increasing the sophistication of those tasks which can be achieved autonomously, it is possible for UAVs to perform more complex missions at lower costs.

This thesis uses a hierarchical decomposition, shown in Fig. 1-5 [1], to subdivide the full UAV planning problem into several sub-tasks. Such a decomposition decouples many of the fundamental planning challenges, allowing each to be pursued independently. In Fig. 1-5, each block in the top row represents a planner subproblem, which receives as input available environmental data and results from the

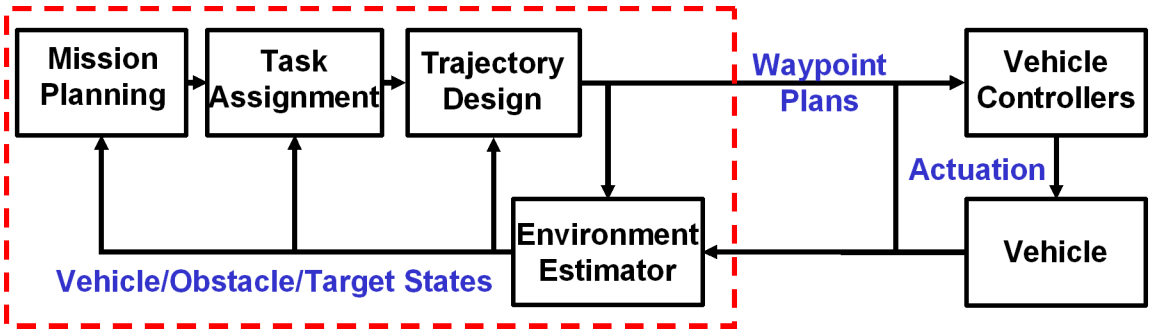


Figure 1-5: Hierarchical decomposition of the multi-UAV planning problem [1]

previous subproblem. The Mission Planning subproblem identifies a set or sequence of tasks which best satisfy some top-level mission objectives, while the Task Assignment subproblem assigns a UAV or sub-team of UAVs to complete each task. The Trajectory Design subproblem uses these task-level waypoints to identify a feasible trajectory waypoint plan for each UAV. Finally, the Vehicle Controllers subproblem includes a trajectory generator to compute an interpolated reference trajectory, as well as a controller which uses deviations from this reference to compute the actuator inputs sent to the vehicle. This thesis focuses on the final two planner tasks in Fig. 1-5, Trajectory Design and Vehicle Controllers.

The objective of this thesis is to develop a UAV trajectory planner to efficiently identify and execute trajectories which are robust to complex, uncertain environments. This planner utilizes disturbance feedback, such that the UAV can safely operate near constraint boundaries without risk of violation. Every trajectory plan identified by the planner includes guaranteed long-term feasibility, regardless of whether additional optimizations are performed. A cost map, approximating the cost to reach the goal from specific points in the environment, allows the UAV to identify intelligent plans with short planning horizons. A variety of implementation refinements are also identified to improve the planner's efficiency.

Using the planner outlined in this thesis, a UAV is capable of autonomously following optimal trajectories, regardless of its initial environmental awareness. The trajectory plans can be designed to be optimal with respect to a variety of performance objectives, such as minimum-time and minimum-fuel criteria. The planner includes

mechanisms which allow the UAV to react to perceived changes in the environment in real-time. A low-level vehicle controller structure which encourages accurate tracking of the optimal waypoint plans is included within this framework.

The work in this thesis exclusively considers the trajectory planning problem for a single vehicle. Because the Task Assignment subproblem assigns task waypoints to each vehicle, a single UAV typically does not need to consider the intentions of other UAVs unless they directly influence its own actions. Previous research [6–10] has considered both centralized and de-centralized trajectory planning for teams of UAVs; however, since the boundary between this work and the Task Assignment subproblem is rather nebulous, it is not considered further here.

1.2 Literature Review

While many techniques are available for addressing the trajectory planning problem [11], this thesis builds upon work in two specific disciplines, robust model predictive control and mixed-integer linear programming. A review of the literature for each field is given below.

1.2.1 Robust Model Predictive Control

Model predictive control (MPC), which originated within the processing industry [12], has received renewed attention in recent decades as advances in computational power have enabled its use for control of systems with faster dynamics [13]. In MPC, a model of predicted system behavior is used to optimize an open-loop, finite-horizon input sequence which satisfies a set of explicit constraints. This optimization may be performed iteratively, in a formulation also known as receding horizon control (RHC). Because constraints are encoded directly into the optimization, MPC-based controllers can operate at or near constraint boundaries while maintaining long-term stability. In the presence of uncertainty, however, these controllers may perform poorly or drive the system out of the feasible region. It is thus important to consider approaches which incorporate knowledge of uncertainty within the optimization,

collectively known as robust model predictive control (RMPC).

In the RMPC formulation, the optimized input sequence must satisfy all constraints subject to an unknown but bounded disturbance sequence. This disturbance is often represented as a parametric uncertainty or additive uncertainty, and may be internal or external to the system. Throughout this thesis, a linear system model subject to additive disturbances is used.

Early work in RMPC identified ways to "robustify" the original MPC formulation by modifying the constraints and objective in the presence of disturbances. Ref. [14] uses constraints on the terminal optimization step and horizon length to guarantee robust feasibility; terminal constraints remain a critical component of most feasibility proofs. Ref. [15] develops a minimax objective function, which identifies the input sequence that minimizes the worst case cost over all possible disturbance sequences. Ref. [16] provides a survey of early RMPC developments.

Though it guarantees robust feasibility, the open-loop RMPC optimization is subject to large regions of infeasibility because it attempts to optimize over every possible disturbance sequence [13]. A more effective approach is to utilize feedback at future optimization steps, as knowledge of the disturbance realization becomes available. It is possible to optimize over arbitrary feedback policies by enumerating all possible worst case disturbance sequences [17]. However, the complexity of such approaches grows exponentially with the problem size, limiting their usefulness in an on-line optimization.

A typical compromise is then to optimize over a specific class of feedback policies. Several parametrizations of feedback policies have emerged in the literature [16], collectively demonstrating an important tradeoff between complexity and conservativeness. In this thesis, the focus is placed on *affine* feedback policies, which combine linear feedback on future states and/or disturbances with a vector of open-loop perturbations. A critical difference among available affine approaches is whether the feedback term is optimized on-line or off-line.

Constraint tightening (CT) policies guarantee robustness by iteratively tightening system constraints *a priori*, retaining margins used to reject future disturbances

with feedback. By selecting the feedback policy off-line, the decision space of the on-line optimization remains the same as the nominal MPC formulation, allowing for efficient computation. The notion of retaining margins for future feedback was originally developed as an input set-aside term [18]. More recent formulations consider both state and input constraints, and expand the original setup to allow constant state feedback [19], time-varying state feedback [20, 21], and time-varying disturbance feedback [22, 23]. The off-line feedback policy can be selected by the user to satisfy various design criteria, such as maintaining large feasibility regions.

Another class of affine RMPC approaches has arisen from the use of linear matrix inequalities (LMI) to frame the robust optimization problem. As with previous minimax formulations [15], LMIs can be used to minimize the worst case cost over all possible disturbance sequences. While early work with LMIs focused on systems with model uncertainty [24], recent literature has identified similar formulations for systems with additive disturbances [25]. Ref. [26] makes the important observation that the optimization over the full affine feedback policy space is convex. The additional degrees of freedom afforded by allowing decision variables in the feedback term can significantly reduce the predicted cost, with the tradeoff of increased problem complexity.

The ideas in Ref. [26] have recently been connected with work in Ref. [27] to create another class of RMPC policies, referred to here as *affine feedback parametrization* (AFP). The AFP approach involves directly inserting the affine feedback policy into both the constraints and the objective function. A variety of objective functions have been considered to complement the worst case LMI of Ref. [26], such as disturbance free cost [28] and expected cost [29]. An important recent theoretical result is the identification of an equivalence relationship between state feedback and disturbance feedback AFP policies, overcoming the difficulties associated with the non-convex policy space for state feedback [28].

This thesis uses a generalized RMPC framework and simulation results to investigate the aforementioned classes of affine feedback policies. While these policies differ significantly in both their derivation and the extent of the on-line optimization,

theoretical results are provided to show that they are actually closely related. A set of numerical simulations are performed to identify the strengths and weaknesses of each approach, with particular emphasis on the tradeoff between complexity and conservativeness.

1.2.2 Mixed-Integer Linear Programming

Mixed-integer linear programming (MILP) provides a very general framework for modeling problems involving both discrete decisions and continuous variables. A linear program is transformed into a MILP by requiring at least one of the decision variables to satisfy an integer (typically binary) value. MILP-based optimization has recently been identified as a suitable candidate for modeling UAV planning problems, which may involve non-convex environments, collision avoidance, waypoint sequences, and other logical and/or temporal constraints [30–32]. MILP can also be applied to control applications such as plume impingement [33], spacecraft formation flight [34], hybrid systems [35–37], and task assignment [6, 38–41].

For trajectory planning problems involving long distances or time intervals, a cost map approximating the cost-to-go from specific points in the environment can be used to design short but intelligent trajectories which help preserve the problem's scalability [42]. One straightforward way to generate this cost map applies a shortest-path algorithm to a visibility graph representation of the environment. For a two-dimensional environment, the graph consists primarily of obstacle vertices [42]; approximate extensions to a three-dimensional environment have also been considered [43]. If appropriate, the cost map can also be modified to ensure that the optimal paths identified through the visibility graph are dynamically feasible [44, 45]. Using this cost-to-go approximation, the planner is capable of identifying local minima "dead ends" within the environment and avoiding them, even if they are beyond the vehicle's current planning horizon.

If a MILP trajectory planner uses short horizon lengths and/or lacks full knowledge of the obstacle environment, it may lead the UAV into dangerous regions where it cannot safely maneuver away. To address this, several papers have identified con-

straints which can be added to the trajectory optimization in order to guarantee vehicle safety. One option is to require the planner to repeatedly generate both the locally optimal solution and a "rescue path" with guaranteed feasibility [46]. Alternatively, the planner can require each plan to end in a terminal basis state where it can remain indefinitely, such as a zero-velocity state for full-stop vehicles [46] or a loiter circle for no-stop vehicles [47]. A side benefit of this guarantee is the classification of the trajectory planner as an anytime algorithm, wherein optimization can be halted at any time without loss of feasibility.

Because constraint tightening can be generalized to a non-convex environment [21, 22], it can also be integrated within the MILP path planning problem. Several variations of this integration have been considered for both single UAVs and teams of UAVs [8, 48, 49]. A recent variation of the planner, the Robust Safe But Knowledgeable (RSBK) algorithm [50], extends this integration to also include the previously-mentioned cost map and guaranteed-safety constraints. A critical component of this algorithm is a parametrization of the terminal invariant set which allows its online selection. A decentralized version of the RSBK algorithm has also been developed [10, 23], based on existing work with decentralized safety algorithms [9].

This thesis aims to develop an autonomous trajectory planner by integrating much of the theoretical work on MILP path planning with several innovations developed to improve implementation efficiency. In particular, this planner, named Efficient RSBK, connects and extends the RSBK algorithm of Ref. [10] and the Variable MILP algorithm of Ref. [51]. The Variable MILP algorithm uses a variety of techniques to enhance optimization efficiency, including variable timestep lengths [52], reachable horizon bounds, linear interpolation points, and non-linear trajectory correction.

1.3 Objectives

The primary objective of this thesis is to design and demonstrate an autonomous UAV system capable of planning and executing robust trajectories which satisfy a collection of success criteria, including safety, optimality, and efficiency. These success criteria

are defined below, after the problem statement is established.

The secondary objective of this thesis is to identify the relative strengths and weaknesses of a collection of RMPC feedback policy types, as well as key theoretical relationships. Because RMPC is a critical component of the robust path planner, the analysis used towards this objective also informs the selection of an appropriate RMPC policy type for the trajectory planner.

1.3.1 Problem Statement

Consider a single UAV operating within a three-dimensional environment. The trajectory planning problem for this UAV is characterized by the following statements:

- The vehicle is modeled as a discrete-time state space model, subject to an unknown but bounded additive disturbance.
- The vehicle's position, $\mathbf{p} \in \mathbb{R}^3$, is required to remain within a set of convex environmental bounds. The vehicle's velocity $\mathbf{v} \in \mathbb{R}^3$ and input acceleration $\mathbf{a} \in \mathbb{R}^m$ are also subject to possibly non-convex constraints, arising from safety considerations and/or hardware limitations.
- The environment may contain a finite number of convex no-fly zones, which are assumed here to be static. Each no-fly zone may be encoded using either hard constraints or soft constraints, depending on its nature.
- The set of no-fly zones may not be fully known by the vehicle at any point in time. However, it is assumed that environmental knowledge is accurate within some *detection radius* R_d of the vehicle. If the vehicle is assumed to have perfect knowledge of its environment, $R_d = +\infty$.
- At any given timestep, the vehicle has been assigned a particular goal waypoint $\mathbf{p}_G$, which may be associated with a goal velocity $\mathbf{v}_G$ and/or other factors, such as an arrival time.
- The objective function may be a combination of several factors, such as min-time terms, state and input weights, and penalties for violating soft constraints.

1.3.2 Success Criteria

Given the problem statement defined in the previous section, the effectiveness of the trajectory planner developed in this thesis is measured in terms of the following success criteria:

1. *All hard constraints are satisfied* by the vehicle at all timesteps and for all possible disturbance sequences. Soft constraints should be satisfied as often as possible, in accordance with the penalties for each in the objective function.

2. The resulting trajectories achieve *good (if not optimal) performance*, as measured by the factors within the objective function.

3. The planner is capable of *real-time operation*, such that the optimization can sufficiently and rapidly respond to environmental changes.

4. The planner *avoids churning*: the qualitative characteristics of the trajectory plan do not change in the absence of new information.

5. The low-level vehicle controller *follows its input waypoint plan* as closely as possible, with particular emphasis placed on arriving at waypoints on time.

1.3.3 Contributions

Each chapter of this thesis offers unique contributions in pursuit of the overall objectives. These contributions are summarized below.

- **Chapter 2**: A comprehensive analysis of affine feedback policies in robust model predictive control is performed, using a generalized RMPC framework and simulation results. Multiple theoretical and numerical relationships between policy types are developed to establish a full characterization of the underlying theory for RMPC affine feedback policies. The strengths and weaknesses of each policy type are identified, as well as some of the fundamental challenges in implementing them. Particular emphasis is placed on the tradeoff between problem complexity and conservativeness.

- **Chapter 3**: A novel MILP-based planner is introduced which is capable of satisfying the first four success criteria defined above. This planner, named Efficient RSBK, connects the RSBK algorithm of Ref. [10] with the Variable MILP algorithm of Ref. [51]. Several additional refinements are proposed, such as Selective CT, variable-density constraint selection, and the use of a detection radius. With these refinements, the planner is shown to maintain robust feasibility under a general disturbance CT policy. These results are presented for both an arbitrary vehicle model and more traditional UAV dynamics. Simulation results demonstrate the effectiveness of the planner.

- **Chapter 4**: A low-level vehicle controller is proposed which significantly improves waypoint-following accuracy compared to previous approaches, satisfying the final success criterion. This vehicle controller includes several refinements, such as spline-based waypoint interpolation, high-order reference tracking, and drag feedforward. The effectiveness of these refinements is demonstrated with quadrotor flights in the RAVEN testbed (Section 4.2). Finally, a fully-integrated Efficient RSBK implementation is demonstrated on RAVEN, satisfying all success criteria on an actual UAV system.

1.4 Approach

This thesis is structured as follows. Chap. 2 performs a theoretical and numerical analysis of RMPC affine feedback policies, and gives results suggesting the strengths of each approach. Chap. 3 introduces the Efficient RSBK trajectory planning algorithm, including several simulation results. Chap. 4 develops the low-level vehicle controller, and demonstrates both the low-level controller and the Efficient RSBK planner through hardware demonstrations on the RAVEN testbed. Finally, Chap. 5 offers concluding remarks and suggestions for future work.

Chapter 2

Affine Feedback Policies in Robust Model Predictive Control

2.1 Introduction

Perhaps the most critical element of a robust trajectory planner is the guarantee of long-term feasibility in the presence of uncertainty. One way to achieve this guarantee is through robust model predictive control (RMPC), in which a model of predicted system behavior is used to optimize a finite-horizon input sequence or policy which satisfies a set of constraints for all possible disturbances. RMPC is an attractive choice for trajectory planning problems, as it can directly encode complex constraints and maintain feasible operation close to constraint boundaries. However, there are many RMPC formulations available for achieving robustness. The appropriate choice for the trajectory planner depends on a variety of factors, such as optimization runtime and cost-based performance.

This chapter uses a generalized RMPC framework and simulation results to investigate two classes of affine feedback policies on the disturbance, constraint tightening (CT) [22] and affine feedback parametrization (AFP) [28]. Policies of this type have been shown to subsume the set of feasible state feedback policies [22] while possessing a convex policy space [28]. However, these two approaches differ significantly in both their derivation and the extent of the on-line optimization. In particular, using a sim-

ple theoretical result, it is shown that these two approaches differ most significantly in the number of decision variables in the on-line optimization. Each decision variable contributes an additional degree of freedom in the optimization, which increases the problem complexity but could potentially improve performance. Both theoretical and simulation results are presented which show that, as expected, these additional decision variables reduce the cost predicted by the optimization objective function.

On the other hand, simulation results are also presented which demonstrate that the closed-loop cost incurred does not always improve with additional decision variables, even when using a conventional terminal cost-to-go and feedback. This leads to the interesting observation that, depending on the problem setup, deviations from the predicted cost may be sufficient to modify the cost ranking among policy types. Furthermore, combined with the computational ranking, this difference may be sufficient to influence the choice between these feedback policies. This is demonstrated on a set of examples for several forms of the objective function, including expected cost [29] and worst case cost via linear matrix inequalities (LMI) [26].

This chapter is structured as follows. Section 2.2 gives the problem statement and the nominal MPC formulation. Section 2.3 reviews the CT and AFP formulations within a generalized framework. After introducing several forms of the objective function in Section 2.4, these formulations are then compared through numerical simulations in Section 2.5. Finally, Section 2.6 offers concluding remarks.

2.2 Preliminaries

2.2.1 Notation

The set $\mathbb{N}_j$ denotes nonnegative integers up to and including j, while the set $\mathbb{N}_{i,j}$ denotes nonnegative integers between i and j inclusive. The symbol I_p denotes the $p \times p$ identity matrix, while the symbol $\mathbf{1}_p$ denotes a p-vector of ones. The operators $\ominus$, $\oplus$, and $\otimes$ denote the Pontryagin difference, Minkowski sum, and Kronecker tensor product, respectively. Finally, $\mathbb{E}$ and tr denote the expectation and trace, respectively.

2.2.2 Problem Statement

Consider the linear time-invariant dynamics with an additive disturbance and output constraints,

$$x_{t+1} = Ax_t + Bu_t + Gw_t, \tag{2.1}$$

$$y_t = Cx_t + Du_t \in \mathbf{S}_y, \tag{2.2}$$

$$w_t \in \mathbf{S}_w, \tag{2.3}$$

where $x_t \in \mathbb{R}^n$ is the state, $u_t \in \mathbb{R}^m$ is the input, $y_t \in \mathbb{R}^p$ is the output, and $w_t \in \mathbb{R}^{n_w}$ is an additive disturbance acting on the state. This disturbance is unknown at current and future timesteps, but is known to fall within the convex and compact set $\mathbf{S}_w$, which contains the origin in its interior. This is a more general form of the additive disturbance than the form used in Refs. [22, 28], in which $n_w = n$ and $G = I_n$. The output is to remain constrained within the polytopic set $\mathbf{S}_y$, which contains the origin in its interior. It is assumed that (A, B) is stabilizable and that the full state is available at all timesteps.

In model predictive control, prediction steps are considered at timestep t for the timesteps t through $t + N$, where N is the user-specified horizon length. It is often useful to concatenate the state, input, disturbance, and output into the vectors

$$\mathbf{x} = \begin{bmatrix} x_{t|t}^T & x_{t+1|t}^T & \cdots & x_{t+N|t}^T \end{bmatrix}^T, \qquad \mathbf{u} = \begin{bmatrix} u_{t|t}^T & u_{t+1|t}^T & \cdots & u_{t+N-1|t}^T \end{bmatrix}^T, \tag{2.4}$$

$$\mathbf{w} = \begin{bmatrix} w_t^T & w_{t+1}^T & \cdots & w_{t+N-1}^T \end{bmatrix}^T, \qquad \mathbf{y} = \begin{bmatrix} y_{t|t}^T & y_{t+1|t}^T & \cdots & y_{t+N-1|t}^T \end{bmatrix}^T, \tag{2.5}$$

respectively, where in $\mathbf{x}$, $\mathbf{u}$, and $\mathbf{y}$, the left subscript is the predicted timestep and the right subscript is the current timestep.

The overall goal using RMPC is to robustly satisfy the constraints (2.2) at all times and for all disturbances $w \in \mathbf{S}_w$, while minimizing the infinite-horizon cost

$$\sum_{t=0}^{\infty} f(x_t, u_t), \tag{2.6}$$

where $f(\cdot,\cdot) \succ 0$. This infinite-horizon RMPC optimization, with feedback, is approximated by the repeated on-line solution of the finite-horizon problem

$$\begin{aligned}
\min_{\mu_j} \quad & J(x_t) = f_N(x_{t+N|t}) + \sum_{j=0}^{N-1} f(x_{t+j|t}, u_{t+j|t}) \\
\text{s.t.} \quad & x_{t+j+1|t} = Ax_{t+j|t} + Bu_{t+j|t} + Gw_{t+j}, \\
& u_{t+j|t} = \mu_j(x_{t|t}, \ldots, x_{t+j|t}, w_t, \ldots, w_{t+j-1}), \\
& y_{t+j|t} = Cx_{t+j|t} + Du_{t+j|t} \in \mathbf{S}_y \ \ \forall\ w_j \in \mathbf{S}_w, \ \ \forall\ j \in \mathbb{N}_{N-1}, \\
& x_{t|t} = x_t, \\
& x_{t+N|t} \in \mathbf{S}_t,
\end{aligned} \tag{2.7}$$

where $f_N(\cdot) \succ 0$ is the approximate infinite-horizon cost-to-go from the terminal state, $\mathbf{S}_t$ is the terminal set, and $\mu_j \ \ \forall\ j \in \mathbb{N}_{N-1}$ is the feedback policy to be identified by the optimization. The feedback policy space is typically constrained to some specific class of feedback policies (Section 2.3). It is assumed that $\mathbf{S}_t$ is polytopic and contains the origin in its interior. Because $\mathbf{S}_y$ and $\mathbf{S}_t$ are polyhedral, they can also be written in the inequality forms

$$\mathbf{S}_y = \{y \in \mathbb{R}^p \mid E_y y \leq f_y\}, \tag{2.8}$$

$$\mathbf{S}_t = \{x \in \mathbb{R}^n \mid E_t x \leq f_t\}, \tag{2.9}$$

where $f_y \in \mathbb{R}^{c_y}$ and $f_t \in \mathbb{R}^{c_t}$.

Model predictive control (MPC) employs the use of a receding horizon, such that the finite-horizon problem (2.7) is solved repeatedly. At each timestep t, the RMPC optimization (2.7) is performed using the current measurement of the state x_t; the first optimized input $u_t = u^*_{t|t}$ is then applied to the system (Algorithm 2.1). The terminal set $\mathbf{S}_t$ is typically chosen by the designer to satisfy certain feasibility and/or convergence criteria, as discussed throughout this chapter.

A primary objective of this analysis is to compare RMPC approaches in terms of both their predicted cost and the actual closed-loop cost incurred. The *incurred*

Algorithm 2.1. Receding Horizon MPC

1: **for** $t = 0$ to $t = \infty$ **do**
2: Take measurement of current state: $x_{t|t} = x_t$
3: Solve MPC optimization (2.7)
4: Apply first input: $u_t = u^*_{t|t}$
5: **end for**

cost at time t is defined as the portion of the infinite-horizon cost (2.6) realized from previous timesteps,

$$J_t = \sum_{j=0}^{t-1} f(x_j, u_j). \tag{2.10}$$

The *predicted cost at time t* includes the estimate of the remaining cost-to-go from the objective function in (2.7), and is defined as

$$J_t^p = J_t + J^\star(x_t). \tag{2.11}$$

In practice, it is most useful to consider the predicted cost at timestep $t = 0$, before any inputs have been applied; this can be written simply as $J^\star(x_0)$ and is hereafter referred to as the *predicted cost*.

2.2.3 Nominal MPC

Because of the unknown disturbances w, the optimization (2.7) cannot be directly implemented as presented. The MPC formulations in this and subsequent sections thus add appropriate restrictions to render the problem tractable. The simplest form of the RMPC optimization (2.7), referred to as the *nominal* formulation, assumes that $w_j \equiv 0 \ \ \forall\, j \in \mathbb{N}_{t,t+N-1}$ and no feedback is applied. However, since this assumption is clearly not true due to (2.1), this naïve approach is *not* robust.

In the absence of disturbances, long-term feasibility can be proven for the nominal receding horizon problem by associating the terminal set $\mathbf{S}_t$ with a terminal control policy $u = \kappa(x)$ which satisfies a set of invariance and feasibility criteria, reviewed below.

Theorem 2.1 (nominal feasibility). *Suppose that* $\mathbf{S}_t$ *and* κ *are defined such that*

$$Ax + B\kappa(x) \in \mathbf{S}_t \ \ \forall\ x \in \mathbf{S}_t, \tag{2.12}$$

$$Cx + D\kappa(x) \in \mathbf{S}_y \ \ \forall\ x \in \mathbf{S}_t. \tag{2.13}$$

If $w_j \equiv 0\ \forall\ j \in \mathbb{N}_\infty$ *and (2.7) is feasible at timestep* $t = 0$*, then a system operating under Algorithm 2.1 satisfies its constraints at all timesteps.*

Proof. Suppose that (2.7) is feasible at timestep t. For the optimization at timestep $t+1$, construct the candidate solution

$$u_{t+j+1|t+1} = u_{t+j+1|t}, \quad \forall\ j \in \mathbb{N}_{0,N-2}, \tag{2.14}$$

$$x_{t+j+1|t+1} = x_{t+j+1|t}, \quad \forall\ j \in \mathbb{N}_{0,N-1}, \tag{2.15}$$

$$u_{t+N|t+1} = \kappa(x_{t+N|t}), \tag{2.16}$$

$$x_{t+N+1|t+1} = Ax_{t+N|t+1} + Bu_{t+N|t+1}. \tag{2.17}$$

In the absence of disturbances, it is clear that (2.14) and (2.15) satisfy the output constraints (2.2) using the final $N-1$ inputs and N states from the previous solution. Since $x_{t+N|t} \in \mathbf{S}_t$, then by (2.12) $x_{t+N+1|t+1} \in \mathbf{S}_t$, and by (2.13) the output constraints (2.2) are satisfied for the final optimization step. Thus the optimization at timestep $t+1$ is feasible. Since (2.7) is assumed to be feasible at $t = 0$, it is therefore feasible for all timesteps. $\diamondsuit$

With proper selection of the cost-to-go $f_N(x_N)$, convergence to the origin in the absence of disturbances can also be proven.

Theorem 2.2 (nominal convergence). *If, in addition to the assumptions on Theorem 2.1,* f_N *satisfies the condition*

$$f_N(Ax + B\kappa(x)) - f_N(x) \leq -f(x, \kappa(x)) \ \ \forall\ x \in \mathbf{S}_t, \tag{2.18}$$

then the system converges asymptotically to the origin.

Proof. The proof utilizes the objective function as a Lyapunov function, and can be found in Refs. [13, 26], among other sources. $\diamondsuit$

2.3 Disturbance RMPC Policies

This section briefly reviews CT policies [18, 19, 21, 22] and AFP policies [28, 29] for RMPC. As stated in Section 2.1, these paradigms use the notion of affine disturbance feedback at future timesteps, written as

$$\mu_j = \sum_{i=0}^{j-1} M_{j,i} w_{t+i} + v_j, \quad \forall\ j \in \mathbb{N}_{N-1}, \tag{2.19}$$

where $M_{j,i}$ is the linear disturbance feedback and v_j is the affine perturbation; the summation bounds are necessary for causality. The key differences between the paradigms are the restrictions on $M_{j,i}$, and whether feedback is computed off-line or on-line. An equivalence relationship between the two approaches is proven in Section 2.3.3.

2.3.1 Constraint Tightening

Constraint tightening policies guarantee robustness by tightening system constraints *a priori*, retaining margins used to reject future disturbances through a feedback policy [18, 19]. Recent research has extended the approach in Ref. [19] to allow time-varying state feedback [21] and disturbance feedback [22], the type reviewed in this section.

In Disturbance CT, the linear disturbance feedback policy $M_j^o\ \forall\ j \in \mathbb{N}_{1,N-1}$ is selected off-line to tighten the constraints $\mathbf{S}_y$ and $\mathbf{S}_t$. The affine perturbations $v_j\ \forall\ j \in \mathbb{N}_{N-1}$ are then optimized as an open-loop input sequence in (2.7). In this manner, the decision space of the on-line optimization remains the same as nominal MPC, allowing for efficient computation.

Because of the tightening, it is necessary to distinguish the output constraints in (2.7) at each step. The output constraints at optimization step j, indicated by $\mathbf{S}_{y|j}$,

are defined by the recursion [22]

$$\begin{aligned} \mathbf{S}_{y|0} &= \mathbf{S}_y, \\ \mathbf{S}_{y|j+1} &= \mathbf{S}_{y|j} \ominus (CL_j + DM_{j+1}^o)\mathbf{S}_w, \quad \forall\ j \in \mathbb{N}_{N-2}, \end{aligned} \tag{2.20}$$

where the L_j are the transition matrix series

$$\begin{aligned} L_0 &= G, \\ L_{j+1} &= AL_j + BM_{j+1}^o, \quad \forall\ j \in \mathbb{N}_{N-2}. \end{aligned} \tag{2.21}$$

Note that $G = I_n$ in Ref. [22]. Additionally, the terminal constraint in (2.7) is replaced with the constraint

$$x_{t+N|t} \in \mathbf{S}_{x|N} = \mathbf{S}_t \ominus L_{N-1}\mathbf{S}_w. \tag{2.22}$$

This tightening recursion remains valid if the convexity assumptions on $\mathbf{S}_t$ and/or $\mathbf{S}_y$ are removed [21]. This is particularly useful in trajectory planning, where obstacles often result in a non-convex physical environment (Chap. 3).

There are several options available for selecting the disturbance policy M_j^o off-line. It has been shown that all state feedback policies can be expressed using disturbance feedback (see Remark 2.1 below). Thus one option is to select the equivalent disturbance form of "useful" state feedback policies, such as LQR or nilpotent policies [21]. Alternatively, because the set of admissible disturbance feedback policies is convex [28], an off-line optimization may be performed to identify an appropriate feedback policy. Ref. [22] proposes an optimization which maximizes the allowable disturbance bounds while maintaining a non-empty terminal set, a necessary condition for a feasible optimization.

Theorem 2.3 (CT robust feasibility). *Suppose* $\mathbf{S}_t$ *and* κ *are defined such that*

$$Ax + B\kappa(x) + L_{N-1}w \in \mathbf{S}_t \ \ \forall\ x \in \mathbf{S}_t, \ \ \forall\ w \in \mathbf{S}_w, \tag{2.23}$$

$$Cx + D\kappa(x) \in \mathbf{S}_{y|N-1} \ \ \forall\ x \in \mathbf{S}_t. \tag{2.24}$$

If (2.7) is feasible at timestep $t = 0$*, then a system operating under Algorithm 2.1 with Disturbance CT satisfies its constraints at all timesteps.*

Proof. This proof closely follows the proof of Theorem 2.1, particularly in the construction of a candidate solution. In this case, the candidate solution takes the form

$$u_{t+j+1|t+1} = u_{t+j+1|t} + M^o_{j+1} w_t, \ \ \forall\ j \in \mathbb{N}_{N-2}, \tag{2.25}$$

$$x_{t+j+1|t+1} = x_{t+j+1|t} + L_j w_t, \ \ \forall\ j \in \mathbb{N}_{N-1}, \tag{2.26}$$

$$u_{t+N|t+1} = \kappa(x_{t+N|t}), \tag{2.27}$$

$$x_{t+N+1|t+1} = Ax_{t+N|t+1} + Bu_{t+N|t+1}. \tag{2.28}$$

The full proof can be found in Ref. [23]. $\diamondsuit$

Remark 2.1. *(relation to previous CT)* Under State CT, a *state* feedback policy of the form $K^o_j \ \forall\ j \in \mathbb{N}_{N-1}$ is applied instead of disturbance feedback [21]. Ref. [22] shows that the set of Disturbance CT policies strictly subsumes the set of State CT policies. In particular, any State CT policy can be rewritten as a Disturbance CT policy through the relation

$$M^o_{j+1} = K^o_j L_j, \ \ \forall\ j \in \mathbb{N}_{N-2}, \tag{2.29}$$

where L_j is defined as in Ref. [22]. Furthermore, the version of State CT in Ref. [21] has been shown to strictly subsume the set of feasible policies in Ref. [19]. These two formulations are thus omitted from this analysis for brevity.

Remark 2.2. *(CT convergence)* No robust convergence theorems have been explicitly demonstrated for Disturbance CT. However, a theorem has been proven for State CT which ensures robust convergence to a target set [21]; based on Remark 2.1, an analog

for Disturbance CT can likely be proven.

2.3.2 Affine Feedback Parametrization

Now consider the AFP approach for RMPC [28], built on the parametrization results from Refs. [26, 27]. In this approach, the feedback law (2.19), written in the stacked form

$$\mathbf{u} = \mathbf{M}\mathbf{w} + \mathbf{v}, \tag{2.30}$$

where $\mathbf{M}$ is lower block triangular, is inserted directly into the optimization (2.7). In contrast to CT, which uses feedback to "pre-stabilize" the system, AFP includes $\mathbf{M}$ in the set of on-line decision variables. The additional degrees of freedom afforded by these decision variables can significantly reduce the predicted cost, with a tradeoff of increased problem complexity.

To apply this feedback law, first note that the constraints in (2.7) can be written in the stacked form

$$\begin{aligned} \mathbf{x} &= \mathbf{A}x_0 + \mathbf{B}\mathbf{u} + \mathbf{G}\mathbf{w}, & (2.31)\\ \mathbf{y} &= \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}, & (2.32)\\ \mathbf{E}_y\mathbf{y} &\leq \mathbf{f}_y \ \ \forall\ \mathbf{w} \in \mathbf{S}_w^N, & (2.33) \end{aligned}$$

where

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \cdots & B \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ G & 0 & \cdots & 0 \\ AG & G & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}G & A^{N-2}G & \cdots & G \end{bmatrix}, \tag{2.34}$$

$$\mathbf{A} = \begin{bmatrix} I_n \\ A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}, \qquad \mathbf{E}_y = \begin{bmatrix} I_N \otimes E_y & 0 \\ 0 & E_t \end{bmatrix}, \qquad \mathbf{f}_y = \begin{bmatrix} \mathbf{1}_N \otimes f_y \\ f_t \end{bmatrix}, \tag{2.35}$$

$\mathbf{C} = I_N \otimes C$, $\mathbf{D} = I_N \otimes D$, and $\mathbf{S}_w^N = \mathbf{S}_w \times \mathbf{S}_w \times \cdots \times \mathbf{S}_w$. Combining (2.30)-(2.33) yields the single vector constraint

$$F\mathbf{v} + (F\mathbf{M} + G)\mathbf{w} \leq \mathbf{f}_y + Hx_0 \ \ \forall \ \mathbf{w} \in \mathbf{S}_w^N, \tag{2.36}$$

where $F = \mathbf{E}_y\mathbf{C}\mathbf{B} + \mathbf{E}_y\mathbf{D}$, $G = \mathbf{E}_y\mathbf{C}\mathbf{G}$, and $H = -\mathbf{E}_y\mathbf{C}\mathbf{A}$. Since this constraint must be satisfied for all disturbance sequences, it must also be satisfied for the worst case disturbance sequence. Thus, without loss of generality, the constraint (2.36) can be rewritten as

$$F\mathbf{v} + \max_{\mathbf{w} \in \mathbf{S}_w^N} (F\mathbf{M} + G)\mathbf{w} \leq \mathbf{f}_y + Hx_0, \tag{2.37}$$

where the maximization is performed row-wise. There are several ways to rewrite this maximization to form a well-posed optimization problem; Ref. [28] details the appropriate steps if $\mathbf{S}_w$ is polyhedral and/or an affine norm-bounded set.

Theorem 2.4 (AFP robust feasibility). *Suppose that $\mathbf{S}_t$ and κ are defined such that*

$$Ax + B\kappa(x) + Gw \ \in \ \mathbf{S}_t \ \ \forall \ x \in \mathbf{S}_t, \ \ \forall \ w \in \mathbf{S}_w \tag{2.38}$$

$$Cx + D\kappa(x) \ \in \ \mathbf{S}_y \ \ \forall \ x \in \mathbf{S}_t. \tag{2.39}$$

If (2.7) is feasible at timestep $t = 0$, then a system operating under Algorithm 2.1 with AFP satisfies its constraints at all timesteps.

Proof. See Ref. [13]. $\diamondsuit$

The presence of $\mathbf{M}$ in the on-line optimization adds $N(N-1)mn_w/2$ decision variables to the problem, and thus can dramatically increase the problem complexity for long horizon lengths. One alternative is to require $\mathbf{M}$ to be block Toeplitz, such that the applied disturbance feedback is time-invariant [26]. This restriction reduces the number of added decision variables to $(N-1)mn_w$, improving the problem scalability while maintaining robust feasibility. In the extreme case, $\mathbf{M}$ can be completely determined off-line, which reduces the problem complexity to that of the nominal case.

Remark 2.3. *(AFP convergence)* Ref. [28] provides a proof of input-to-state stability, which implies that the system converges to the origin if the initial state is feasible and the disturbance w is either equivalent to or approaching zero over time. Furthermore, because $\mathbf{S}_w$ is bounded, boundedness of the resulting state sequence can also be guaranteed. Note that (2.18) is the only additional assumption necessary for this property.

2.3.3 Equivalence

In (2.37), the maximization term operates on the product of $(F\mathbf{M}+G)$, a lower block triangular matrix, and the disturbance $\mathbf{w}$. Since lower rows in this constraint correspond to later timesteps, this form suggests that some type of disturbance tightening takes place in AFP. The following proposition confirms that, under appropriate assumptions on $\mathbf{M}$ and $\mathbf{S}_{x|N}$, AFP and CT policies are equivalent.

Proposition 2.5 (CT-AFP equivalence). *Suppose $\mathbf{S}_{x|N}$ is given. Then the following two policies are equivalent in the sense that they yield the same solution:*

- *Disturbance CT, using disturbance feedback policy M_j^o $\forall\ i \in \mathbb{N}_{1,N-1}$*
- *AFP, with $\mathbf{M}$ fixed off-line according to the relation*

$$M_{j,i} = M_{j-i}^o \quad \forall\ j \in \mathbb{N}_{1,N-1}, \quad \forall\ i \in \mathbb{N}_{j-1}$$

Proof. In (2.20), each subsequent tightened set is formed by a Pontryagin difference between the previous set and the image of $\mathbf{S}_w$ under a matrix. Since the image of a polyhedral set and the Pontryagin difference between polyhedral sets are both polyhedral [53], the subsequent sets can be characterized as

$$\begin{aligned}\mathbf{S}_{y|j} &= \{y \in \mathbb{R}^p \mid E_j y \leq f_j\} \ \ \forall\ j \in \mathbb{N}_{N-1} \\ &= \{y \in \mathbb{R}^p \mid e_{ji}^T y \leq f_{ji}, \ \ \forall\ i \in \mathbb{N}_{c_y^j}\} \ \ \forall\ j \in \mathbb{N}_{N-1},\end{aligned}$$

where e_{ji}^T is the ith row of E_j and f_{ji} is the ith element of f_j. Substituting the explicit form of (2.21) into (2.20) yields $\mathbf{S}_{y|j+1} = \mathbf{S}_{y|j} \ominus \tilde{L}_j \mathbf{S}_w$, where

$$\tilde{L}_j = CA^j + DM_{j+1}^o + \sum_{i=1}^{j} CA^{j-i} B M_i^o.$$

Let $h_U(v)$ denote the support function of U at vector v. Applying Theorem 2.3 and the properties of the support function in Ref. [53], as well as the fact that $\mathbf{S}_w$ is compact:

$$\begin{aligned}\mathbf{S}_{y|j+1} &= \{y \in \mathbb{R}^p \mid e_{ji}^T y \leq f_{ji} - h_{\tilde{L}_j \mathbf{S}_w}(e_{ji}), \ i \in \mathbb{N}_{c_y}\} \\ &= \{y \in \mathbb{R}^p \mid e_{ji}^T y \leq f_{ji} - h_{\mathbf{S}_w}(\tilde{L}_j^T e_{ji}), \ i \in \mathbb{N}_{c_y}\} \\ &= \{y \in \mathbb{R}^p \mid e_{ji}^T y \leq f_{ji} - \sup_{w \in \mathbf{S}_w} (e_{ji} \tilde{L}_j w), \ i \in \mathbb{N}_{c_y}\} \\ &= \left\{y \in \mathbb{R}^p \mid E_j y \leq f_j - \max_{w \in \mathbf{S}_w} E_j \tilde{L}_j w\right\}.\end{aligned}$$

Thus $E_j = E_y \ \forall j \in \mathbb{N}_{N-1}$ and

$$f_{j+1} = f_j - \max_{w \in \mathbf{S}_w} E_y \tilde{L}_j w.$$

By defining the block Toeplitz matrix

$$\Lambda = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ F_0 & 0 & \cdots & 0 \\ F_1 & F_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ F_{N-1} & F_{N-2} & \cdots & F_0 \end{bmatrix},$$

where $F_j = E_y \tilde{L}_j$, the CT constraints (2.20) can be written in the form

$$F\mathbf{u} \leq \mathbf{f}_y + Hx_0 - \max_{\mathbf{w} \in \mathbf{S}_w^N} \Lambda \mathbf{w},$$

where some notation from (2.36) is used. It is straightforward to show that $\Lambda = F\mathbf{M} + G$ from (2.36) if $M_{j,i} = M^o_{j-i} \quad \forall\ j \in \mathbb{N}_{1,N-1}, \quad \forall\ i \in \mathbb{N}_{j-1}$, verifying the equivalence. $\diamondsuit$

Proposition 2.5 is only valid when the same *tightened* terminal set $\mathbf{S}_{x|N}$ is used for each policy (Section 2.5.2). Because the criteria for robust feasibility differ for each approach, this may not be the case, in general.

2.4 Objective Function

Because the previous robust feasibility and convergence results depend only on a general form of the objective function, the user maintains some freedom in selecting a formulation appropriate for the problem at hand. However, the objective function as presented in (2.7) is inadmissible, since future states depend on a sequence of disturbances which is not known at the time of the optimization. In this section, three formulations which remove dependence on an unknown w are considered: assuming $w = 0$ within the objective function, taking the expectation over w, or considering the worst case.

In this and subsequent sections, the objective function is chosen to be quadratic

in the state and input, such that

$$f_N(x) = (x - x_G)^T P (x - x_G), \tag{2.40}$$

$$f(x,u) = (x - x_G)^T Q (x - x_G) + (u - u_G)^T R (u - u_G), \tag{2.41}$$

where $P,\ Q,\ R \succ 0$, and x_G and u_G represent an equilibrium state and input to be tracked, respectively. Inserting (2.40)-(2.41) along with (2.30)-(2.31) into the objective function of (2.7) yields the stacked cost

$$\begin{aligned} J(x_t) = {} & (\mathbf{A}x_0 + \mathbf{B}\mathbf{M}\mathbf{w} + \mathbf{B}\mathbf{v} + \mathbf{G}\mathbf{w} - \mathbf{x}_G)^T \mathbf{Q} (\mathbf{A}x_0 + \mathbf{B}\mathbf{M}\mathbf{w} + \mathbf{B}\mathbf{v} + \mathbf{G}\mathbf{w} - \mathbf{x}_G) \\ & + (\mathbf{M}\mathbf{w} + \mathbf{v} - \mathbf{u}_G)^T \mathbf{R} (\mathbf{M}\mathbf{w} + \mathbf{v} - \mathbf{u}_G), \end{aligned} \tag{2.42}$$

where $\mathbf{x}_G = \mathbf{1}_{N+1} \otimes x_G$, $\mathbf{u}_G = \mathbf{1}_N \otimes u_G$, $\mathbf{Q} = \begin{bmatrix} I_N \otimes Q & 0 \\ 0 & P \end{bmatrix}$, and $\mathbf{R} = I_N \otimes R$. (Note that (2.30) can be used to represent either RMPC policy type, in accordance with Proposition 2.5.) The *linear* terminal control law $u = \kappa(x) = Kx$ is selected off-line by the designer, and should be stabilizing. The terminal cost-to-go matrix P is found using the steady-state relation for unconstrained linear control,

$$P = Q + K^T R K + (A + BK)^T P (A + BK). \tag{2.43}$$

For more information on P and K, see Section 2.5.2.

2.4.1 Disturbance Free Cost

The simplest form of the objective function disregards the disturbances *in the objective function*, i.e. $\mathbf{w} \equiv 0$. Applying $\mathbf{w} \equiv 0$ to (2.42) results in the "optimistic" objective function used in Refs. [22, 28],

$$\begin{aligned} J_{\mathrm{df}}(x_t) = {} & (\mathbf{A}x_0 + \mathbf{B}\mathbf{v} - \mathbf{x}_G)^T \mathbf{Q} (\mathbf{A}x_0 + \mathbf{B}\mathbf{v} - \mathbf{x}_G) \\ & + (\mathbf{v} - \mathbf{u}_G)^T \mathbf{R} (\mathbf{v} - \mathbf{u}_G). \end{aligned} \tag{2.44}$$

This approach is still robustly feasible, since the disturbances remain accounted for in the constraints.

2.4.2 Expected Cost

Ref. [29] considers an alternative cost formulation which takes the *expected* cost over all possible disturbances, i.e.

$$J_{\text{ex}}(x_t) = \mathbb{E}_{\mathbf{w}\in\mathbf{S}_w}[J(x_t)]. \tag{2.45}$$

Assume that the disturbances w are independent and identically distributed between timesteps, with $E[w] = 0$ and $E[ww^T] = C_w$; by assuming the disturbance to be zero-mean, all terms linear in $\mathbf{w}$ in (2.42) evaluate to zero under the expectation. Indeed, it is straightforward to show that the expected objective function in this case is simply the disturbance free cost (2.44) plus several quadratic disturbance terms,

$$\begin{aligned} J_{\text{ex}}(x_t) &= J_{\text{df}}(x_t) + \text{tr}(\mathbf{M}^T(\mathbf{B}^T\mathbf{Q}\mathbf{B} + \mathbf{R})\mathbf{M}\mathbf{C}_w) \\ &\quad + \text{tr}(2\mathbf{C}_w\mathbf{G}^T\mathbf{Q}\mathbf{B}\mathbf{M}) + \text{tr}(\mathbf{G}^T\mathbf{Q}\mathbf{G}\mathbf{C}_w), \end{aligned} \tag{2.46}$$

where $\mathbf{C}_w = I_N \otimes C_w$. Despite the apparent complexity of (2.46), it is still a quadratic cost function. Note that the only possible decision variables in the additional terms arise through $\mathbf{M}$; thus the disturbance free cost solutions and expected cost solutions are equivalent for those policies in which $\mathbf{M}$ is fixed off-line. For those policies in which $\mathbf{M}$ is not fixed, it may be possible to select a new policy which decreases the overall expected cost.

2.4.3 Worst Case Cost

A cost metric extensively researched in the literature considers the minimization of the *worst case* cost over all possible disturbances. The ideal approach would be to

optimize with the arrival of each new measurement, such that [17]

$$J_{\text{wc}}(x_t) = \min_{u_{t|t}} \max_{w_t} \cdots \min_{u_{t+N-1|t}} \max_{w_{t+N-1}} J(x_t), \tag{2.47}$$

but such an approach is computationally intensive, at best. A tractable alternative considered in Ref. [26] groups the inputs and disturbances, such that

$$J_{\text{wc}}(x_t) = \min_{\mathbf{u}} \max_{\mathbf{w}} J(x_t). \tag{2.48}$$

By applying an epigraph,

$$\begin{aligned} J_{\text{wc}}(x_t) \quad &= \quad \min_{\mathbf{u}} \tau_0, \\ &\text{s.t.} \quad J(x_t) \leq \tau_0, \end{aligned} \tag{2.49}$$

the original objective function can be converted into an LMI, as is common in many robust optimization problems [24]. It can be shown (see Ref. [26] for details) that the LMI for the cost (2.42) can be written as

$$\begin{bmatrix} \tau_0 - \sum_{i=1}^{n_w N} \tau_i & (\star) & (\star) & (\star) \\ \mathbf{A}x_0 + \mathbf{B}\mathbf{v} - \mathbf{x}_G & \mathbf{Q}^{-1} & (\star) & (\star) \\ \mathbf{v} - \mathbf{u}_G & 0 & \mathbf{R}^{-1} & (\star) \\ 0 & (\mathbf{B}\mathbf{M} + \mathbf{G})^T & \mathbf{M}^T & \operatorname{diag} \tau_i \end{bmatrix} \succeq 0, \tag{2.50}$$

subject to the additional constraints $\tau_i \geq 0 \;\; \forall \; i \in \mathbb{N}_{1,n_w N}$. Here the diag operator indicates the placement of the τ_i scalars along the diagonal, while $(\star)$ represents the appropriate term for symmetry. Because of the LMI (2.50), a semidefinite optimization must be performed. Furthermore, $n_w N+1$ decision variables and $n_w N$ inequality constraints are added to the optimization through these constraints.

2.5 Analysis and Simulation

In this section, theoretical and simulation results are presented to investigate the relationship between CT and AFP policies, with emphasis on complexity and the cost metrics noted in Section 2.2.2. After introducing the numerical example, the terminal sets are compared, followed by the cost and complexity. From Proposition 2.5, it is clear that the RMPC policies considered here are distinguished only by the constraints on $\mathbf{S}_{x|N}$ and the extent to which $\mathbf{M}$ is allowed to be adjusted in the on-line optimization. As a result, a key distinguishing feature of these policies is the number of on-line decision variables, a point emphasized throughout this section.

Though the simulation results in this section consider only a few specific examples, only a single example is necessary to demonstrate the main result of this section: the predicted cost may not be an accurate predictor of actual closed-loop performance, in general. The simulations use conventional forms of the terminal set, cost-to-go, and control law found in the literature, as well as a relatively large prediction horizon. Similar conclusions can be drawn from other realistic examples, possibly involving more states/inputs or other additional complexities.

Nine MPC controllers are considered in the simulation results which follow, for a variety of objective functions and disturbance realizations. Note that *all* approaches use K_{nil}, defined in Section 2.5.2, as the terminal control law $\kappa(x) = Kx$.

- **Nominal:** nominal MPC; disturbances are present but ignored
- **CT-LQR:** Dist. CT, using constant state feedback policy K_{lqr}
- **CT-Nil:** Dist. CT, using constant state feedback policy K_{nil}
- **CT-Opt:** Dist. CT, using the off-line optimization in Ref. [22] with $s = 5$
- **AFP-Full:** AFP, with no additional restrictions
- **AFP-BT:** AFP, with the restriction that $\mathbf{M}$ be block Toeplitz
- **AFP-Fix-LQR:** AFP, with the restriction that $\mathbf{M}$ be fixed to equivalent policy of CT-LQR

- **AFP-Fix-Nil:** AFP, with the restriction that $\mathbf{M}$ be fixed to equivalent policy of CT-Nil
- **AFP-Fix-Opt:** AFP, with the restriction that $\mathbf{M}$ be fixed to equivalent policy of CT-Opt

2.5.1 Numerical Example

(*Note:* The example presented here is heavily based on the setup in Section 2.5.1 of Ref. [20].)

Consider the discrete model for a one-dimensional double integrator system ($dt = 1$ second) with an additive disturbance acting on the input acceleration,

$$x_{t+1} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x_t + \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} u_t + \gamma \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} w_t, \tag{2.51}$$

where $x_t = \begin{bmatrix} p_t & v_t \end{bmatrix}^T$ is the state (p_t is the position; v_t is the velocity), u_t is the (acceleration) input, and w_t is the additive disturbance, which is unknown *a priori* but guaranteed to fall within the ∞-norm-bounded set

$$\mathbf{S}_w = \{w \mid \|w\|_\infty \leq 1\}. \tag{2.52}$$

The quantity γ is the disturbance level, here set to 0.3. Using an acceleration not to exceed a magnitude of 1, the system is to satisfy unity constraints on the position and velocity magnitude:

$$y_t = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} x_t + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u_t \in \mathbf{S}_y, \tag{2.53}$$

$$\mathbf{S}_y = \{y \mid \|y\|_\infty \leq 1\}. \tag{2.54}$$

The system begins at the initial condition $x_0 = \begin{bmatrix} 0.5 & 0.5 \end{bmatrix}^T$, a position of +0.5

meters and a velocity of +0.5 m/s. The objective is to regulate the system about the origin, which is a zero-input equilibrium point: $(x_G, u_G) = \mathbf{0}$.

The MPC optimization uses a horizon length N of 10, along with respective state and input quadratic costs

$$Q = \frac{1}{1000}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad R = 100.$$

These costs correspond to a relatively large weight on the acceleration, compared to the position and velocity. The objective function may use a disturbance free cost, an expected cost, or a worst case cost (Section 2.4).

Each comparison in this section averages results from 20 simulations, with the same disturbance sequence applied to every RMPC controller in each. Each simulation consists of 50 consecutive applications of Algorithm 2.1, executed once per timestep. The disturbance realization calculated for the simulated actual system response is distributed either uniformly within $\mathbf{S}_w$ or among the vertices of $\mathbf{S}_w$, i.e. +1 and -1. If the disturbances are uniformly distributed, the mean is 0 and the variance is 1/3 ($C_w = 1/3$). If the disturbances are vertex-only, then the mean is 0 and the variance is 1 ($C_w = 1$).

2.5.2 Terminal Sets

In a receding horizon implementation, the terminal set must be chosen based on criteria which guarantee robust feasibility, and possibly robust convergence. However, with these criteria in place, the designer is free to choose any terminal set which satisfies these criteria. This section explores the possible terminal sets for the system in Section 2.5.1, based on the criteria provided in Sections 2.3.1 – 2.3.2.

While the criteria for robust feasibility vary with each approach, they typically require invariance and feasibility criteria on the terminal set $\mathbf{S}_t$ and terminal control law $u = Kx$ [13]. Since the terminal control law has been assumed to be linear, the

CT robust feasibility criteria (2.22)-(2.24) can be simplified as

$$\begin{aligned}(A+BK)x + L_{N-1}w &\in \mathbf{S}_t \ \ \forall\ w \in \mathbf{S}_w, \ \ \forall\ x \in \mathbf{S}_t, && (2.55)\\ (C+DK)x &\in \mathbf{S}_{y|N-1} \ \ \forall\ x \in \mathbf{S}_t, && (2.56)\\ \mathbf{S}_{x|N} &= \mathbf{S}_t \ominus L_{N-1}\mathbf{S}_w; && (2.57)\end{aligned}$$

recall that $\mathbf{S}_{x|N}$ denotes the tightened terminal constraints. Similarly, the appropriate AFP criteria (2.38)-(2.39) under a linear terminal control law are

$$\begin{aligned}(A+BK)x + Gw &\in \mathbf{S}_t \ \ \forall\ w \in \mathbf{S}_w, \ \ \forall\ x \in \mathbf{S}_t, && (2.58)\\ (C+DK)x &\in \mathbf{S}_y \ \ \forall\ x \in \mathbf{S}_t, && (2.59)\\ \mathbf{S}_{x|N} &= \mathbf{S}_t \ominus L_0\mathbf{S}_w \ominus \cdots \ominus L_{N-1}\mathbf{S}_w. && (2.60)\end{aligned}$$

Note that the transition matrices L_i used in (2.60) are determined on-line for AFP-Full and AFP-BT. Neither of the resulting tightened terminal sets $\mathbf{S}_{x|N}$ is more restrictive for all problem setups, given the same feedback policy. In particular, (2.56) is more restrictive than (2.59), but (2.59) and (2.60) are more restrictive than (2.55) and (2.57), respectively.

Two choices are considered here for the terminal cost-to-go matrix P and control law $\kappa(x) = Kx$. One possible choice is the discrete LQR solution for (A, B, Q, R), $K = K_{lqr} \equiv \begin{bmatrix} -0.0030 & -0.0780 \end{bmatrix}$. However, this terminal control law yields an infeasible terminal set for every RMPC policy type being considered, and thus is not used. Instead, the following simulation results use the 2-step nilpotent policy suggested in Ref. [20], $K = K_{nil} \equiv \begin{bmatrix} -1 & -1.5 \end{bmatrix}$. Its corresponding terminal cost-to-go is found using the steady-state relation (2.43); for this problem,

$$P_{nil} = \begin{bmatrix} 200.0 & 200.0 \\ 200.0 & 250.0 \end{bmatrix}.$$

All simulation results which follow use $K = K_{nil}$ and $P = P_{nil}$.

Even using K_{nil} as the terminal controller, several of the RMPC policy types being

considered are not feasible for this setup. The CT-LQR policy does not have a feasible terminal set, due to poor disturbance attenuation in (2.55) and significant constraint tightening in (2.56). Furthermore, the AFP-Fix-LQR and AFP-Fix-Opt policies, despite having a feasible terminal set, do not have a feasible initial optimization. Since the first optimization must be feasible for robust feasibility, this excludes these approaches from further consideration. This leaves six remaining feasible policies: Nominal (MPC), CT-Nil, CT-Opt, AFP-Full, AFP-BT, and AFP-Fix-Nil.

Given a persistent disturbance, any cost-to-go of the form (2.40) is likely to be a poor approximation of the true cost-to-go. Nonetheless, it is useful in satisfying stability criteria such as (2.18), and is the conventional choice in the literature. More complex forms of the cost-to-go have been considered to improve this approximation; see the Appendix for details.

The terminal set $\mathbf{S}_t$ for each policy type is chosen to be the maximal robust control invariant (RCI) set using $K = K_{nil}$ and satisfying that policy's corresponding feasibility criteria. In the nominal MPC case, only control invariance is necessary. The terminal set center (x_V, u_V) is chosen to coincide with the origin. This is appropriate for this particular problem, since it coincides with the goal (x_G, u_G), and such coincidence is typically necessary to prove robust stability or recover stability in the absence of disturbances.

2.5.3 Cost

The following results compare the predicted cost and average incurred cost, as defined in Section 2.2, of each MPC and RMPC policy using a disturbance free objective (Section 2.4.1), expected objective (Section 2.4.2), or worst case objective (Section 2.4.3). The disturbance realization is either uniformly distributed in $\mathbf{S}_w$ or constrained to the vertices of $\mathbf{S}_w$. Since the type of objective function and disturbance realization both influence the optimization itself, a separate simulation must be performed for each selection. This implementation uses the CLP solver [54] for quadratic programs and the SeDuMi solver [55] for semidefinite programs through the YALMIP interface [56], as well as functionality from several toolboxes [57, 58].

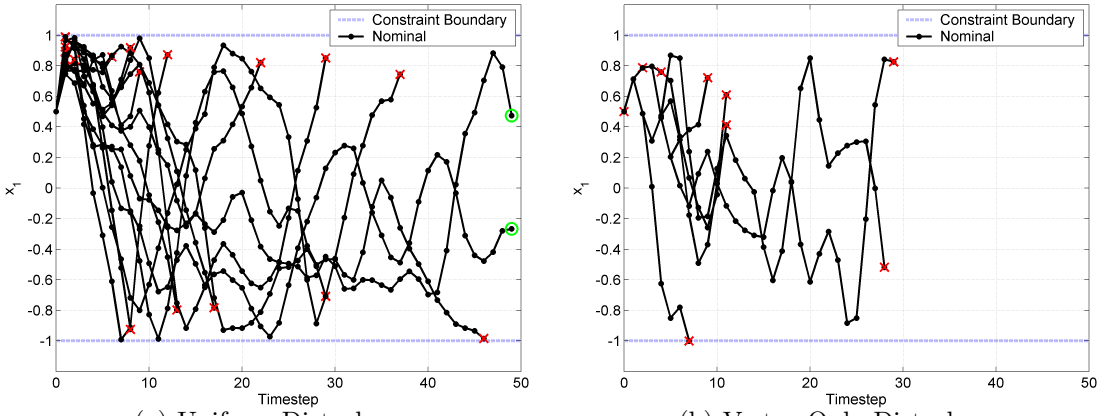
For a single optimization, relationships between the objective function, and hence the predicted cost, for each approach can be constructed based on the constraints and the resulting size of the policy space. For example, the relationship

$$J^\star(x_t)^{\text{AFP-Fix-Nil}} \geq J^\star(x_t)^{\text{AFP-BT}} \geq J^\star(x_t)^{\text{AFP-Full}} \tag{2.61}$$

is valid for each form of the objective function, since each policy is more restrictive moving from right to left. This is a simple consequence of the more general statement that if two optimizations differ *only* in the number of available free decision variables, the optimization with more decision variables should identify an equal or lower optimal cost.

This section makes similar comparisons for the entire collection of RMPC approaches being considered, which differ most significantly in the number of available on-line decision variables. Proposition 2.5 has demonstrated that, aside from differences in $\mathbf{S}_{x|N}$, each policy type differs only in the number of on-line decision variables allowed through $\mathbf{M}$. When the number of on-line decision variables is increased, there is a near-certain resulting increase in computational complexity (Section 2.5.4). It is then a reasonable hypothesis that the RMPC approaches with more decision variables (AFP-Full, then AFP-BT) should be able to "trade off" this increased computation time to achieve better performance (i.e. lower costs) than the approaches with fewer decision variables (CT-Nil, CT-Opt, and AFP-Fix-Nil). This is especially valid for the predicted cost, which is identified through a single optimization.

For the incurred cost, the situation is not quite as clear, since this cost is the summation of terms collected from multiple optimizations through Algorithm 2.1. Nonetheless, it is reasonable to hypothesize that the relationships for the predicted cost also carry over the incurred cost. In particular, increasing the number of on-line decision variables should lead to a trade-off in the increased computational effort versus a decrease in the incurred cost, according to this hypothesis. However, the results that follow demonstrate that this hypothesis, referred to hereafter as the decision variable hypothesis (DVH) for brevity, is not always correct.



(a) Uniform Disturbance (b) Vertex-Only Disturbance

Figure 2-1: Sample position trajectories using Nominal MPC ('X' indicates loss of feasibility).

Nominal MPC

Because the Nominal MPC formulation performs particularly poorly compared to the RMPC formulations, it is considered separately from the main results. Note that because the Nominal MPC formulation ignores disturbances completely, the choice of objective function is irrelevant; it always yields the same result subject to the same disturbance realization.

When subject to a uniform disturbance realization (Fig. 2-1(a)), the Nominal MPC formulation goes infeasible during 18 of the 20 simulations, while *all* simulations go infeasible for the vertex-only disturbance realization (Fig. 2-1(b)). Nominal MPC performs poorly in this scenario primarily due to a combination of two factors. First, the Q/R ratio is sufficiently small that the MPC optimization allows significant state deviations in order to minimize input usage. Second, because Nominal MPC ignores disturbances, the optimization is not prevented from selecting trajectories which fall on or near constraint boundaries. For this problem, the limiting constraint is the position p_t boundary; the MPC optimization will occasionally select plans where one or two optimization steps fall directly on this boundary. Such plans clearly have a high risk of becoming infeasible if an aggravating disturbance is realized.

The Nominal MPC formulation's behavior in the initial timesteps of each simula-

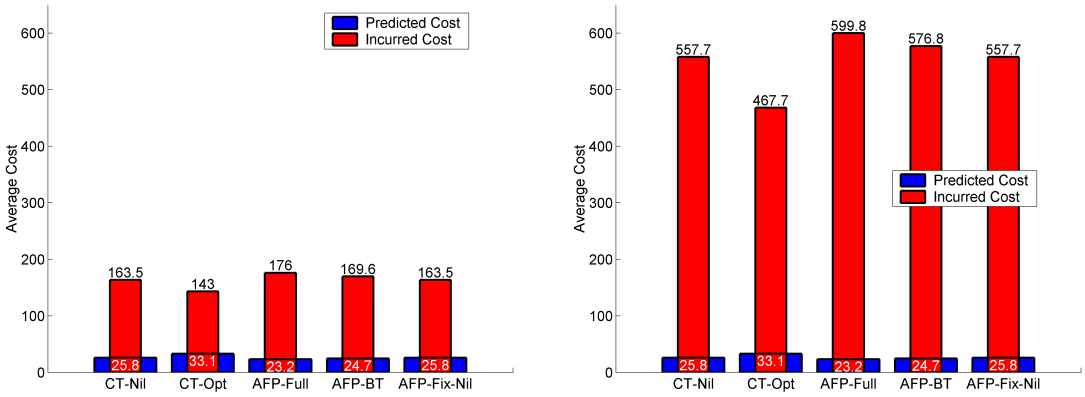
tion is particularly revealing . Recall that the system begins with an initial position of +0.5 m and velocity of +0.5 m/s; gone unchecked, the system will bypass the upper position boundary after 1 timestep. In the optimization at the first timestep, the Nominal MPC formulation selects a weaker corrective input (-0.27) than the RMPC approaches (between -0.55 and -0.44), which maintains a "buffer" on the constraint boundary to ensure robust feasibility. Under the uniform disturbance realization (Fig. 2-1(a)), five of the Nominal MPC simulations become infeasible after two timesteps, due to consecutive positive disturbances. Under the vertex-only disturbance realization (Fig. 2-1(b)), 11 Nominal MPC simulations become infeasible after just one timestep, since in those cases the first disturbance realized is +1.0. Nearly all of the other simulations become infeasible at later timesteps due to similar circumstances.

It is worth noting that the two Nominal MPC simulations which do remain feasible, indicated by a green 'O' in Fig. 2-1(a), by far have the lowest predicted and incurred costs compared to all RMPC approaches considered. Nominal MPC also has a significant runtime advantage under the worst case objective function, since it does not need the computationally expensive LMI setup (Section 2.4.3). By ignoring disturbances, Nominal MPC is the least conservative approach - but to the extent that it is rarely feasible. Clearly, robust MPC must be applied to this problem for successful operation, rather than MPC or classical control schemes.

Disturbance Free Objective

Fig. 2-2 compares the predicted cost $J^*(x_0)$ and the incurred cost J_{50}, averaged over 20 simulations, for each policy type using a disturbance free objective function. Fig. 2-2(a) shows results for a uniform disturbance realization, while Fig. 2-2(b) shows results for a vertex-only disturbance realization. Note that *all* RMPC simulations from this point forward maintain robust feasibility at all timesteps, as expected.

From Fig. 2-2, it is clear that the incurred cost is significantly larger than the predicted cost for all policies, especially for the vertex-only disturbance realization. This suggests, as discussed in Section 2.5.2, that the cost-to-go term $f_N(x_N)$ does not accurately predict the future incurred cost. This is hardly surprising, since the



(a) Uniform Disturbance

(b) Vertex-Only Disturbance

Figure 2-2: Minimization of Disturbance Free Cost

quadratic cost-to-go ignores constraints *and* all future disturbances.

The DVH is accurate for the predicted costs for both disturbance realizations. (Because the objective function ignores disturbances, the predicted costs are the same for each policy for both types of disturbance realization.) In both instances, AFP-Full has the lowest predicted cost, followed by AFP-BT, CT-Nil/AFP-Fix-Nil, and finally CT-Opt. This is consistent with the number of on-line decision variables, as AFP-Full has the most, followed by AFP-BT and the nominal-complexity CT policies. Finally, note that CT-Nil and AFP-Fix-Nil achieve equal predicted costs. Both approaches share the same feedback policy by construction, and because K_{nil} is nilpotent, the terminal constraints (2.55)-(2.57) and (2.58)-(2.60) are equivalent. Since these are the only two factors distinguishing any of these RMPC policies, CT-Nil and AFP-Fix-Nil are equivalent throughout these simulation results.

To see why this predicted cost ranking occurs, recall that the objective function for this problem (ignoring the extremely small state costs) is

$$J^*(x_0) \approx 200p_{N|0}^2 + 400p_{N|0}v_{N|0} + 250v_{N|0}^2 + \sum_{j=0}^{N-1} 100u_{j|0}^2.$$

The terminal cost terms are a very small portion ($< 10\%$) of the total predicted cost for every RMPC approach, as they all drive the state close to zero by the final

Table 2.1: Predicted Cost, First Two Inputs (Disturbance Free Objective)

RMPC Type	$u_{0\|0}$	$u_{1\|0}$	$100(u_{0\|0}^2 + u_{1\|0}^2)$
CT-Nil	-0.482	-0.153	25.6
CT-Opt	-0.549	-0.153	32.5
AFP-Full	-0.443	-0.184	23.0
AFP-BT	-0.471	-0.154	24.5
AFP-Fix-Nil	-0.482	-0.153	25.6

optimization step. Indeed, almost the entire predicted cost is concentrated into the first two optimization inputs, when the system is trying to recover from a poor initial condition. The values of $u_{0|0}$ and $u_{1|0}$, as well as their associated cost (compare with Fig. 2-2(a)), are given in Table 2.1. AFP-Full, with the most decision variables, does the best job of distributing the necessary initial input to stay feasible while minimizing the cost. CT-Opt, whose off-line optimization is designed to maximize disturbance handling, *not* predicted cost, has the highest predicted cost for these terms.

On the other hand, the DVH is *not* accurate for the incurred cost for both disturbance realizations. In fact, the order is completely inverted: CT-Opt has the lowest incurred cost, followed by CT-Nil/AFP-Fix-Nil, AFP-BT, then AFP-Full. This ranking directly contradicts the DVH, which suggests that AFP-Full might have the lowest cost, followed by AFP-BT.

To demonstrate why the incurred cost ranking becomes inverted, consider Fig. 2-3, which shows the position and velocity history for each RMPC controller for one of the simulations; other simulations are similar in form. Because the AFP controllers have more available on-line decision variables than CT, they are able to identify trajectory plans which come closer to the constraint boundaries while maintaining robust feasibility. The AFP controllers choose to execute these trajectories in an effort to minimize control usage, the dominant term in the objective function. However, because the objective function ignores disturbances, the AFP trajectories do not plan ahead for the inputs which may accrue from reacting to unforseen disturbances. While this also affects the CT controllers, they tend to stay closer to the origin, and thus require less reactive input to correct for aggravating disturbances. While this does not occur for every aggravating disturbance, the long-term results average towards

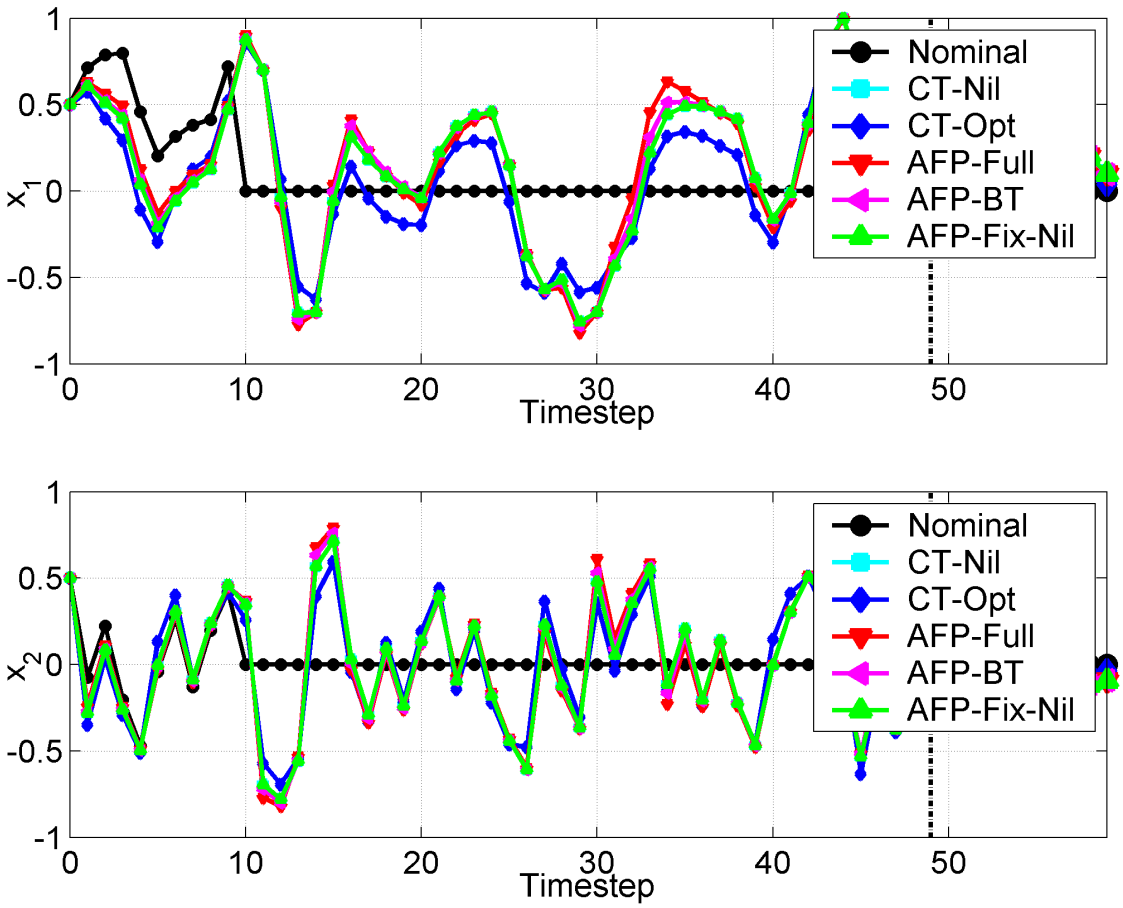


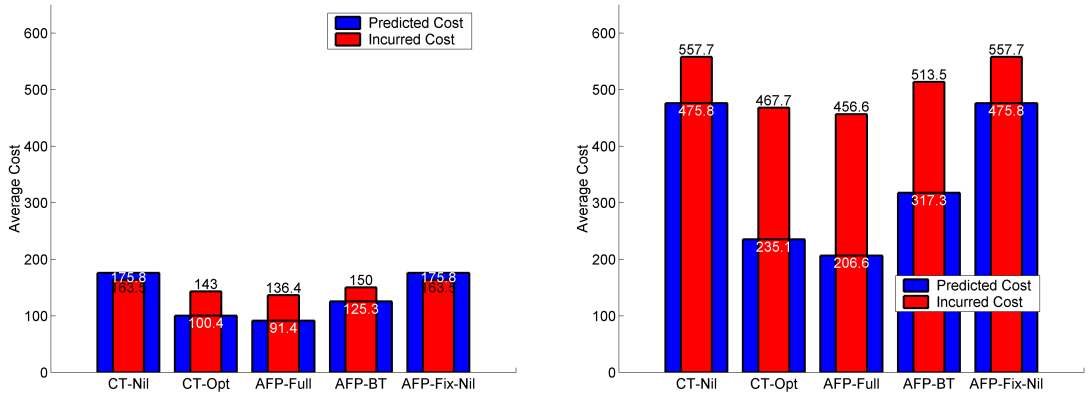
Figure 2-3: Sample state history and current state plan for each RMPC approach after 50 timesteps have been completed, using a disturbance free objective and uniform disturbance realization. The nominal policy becomes infeasible at timestep 10.

lower incurred costs for the CT policies. Clearly, these deviations can be sufficient in general to influence the selection of a cost-minimizing RMPC policy, especially considering differences in computational complexity (Section 2.5.4).

Expected Objective

Fig. 2-4 compares the predicted cost $J^*(x_0)$ and the incurred cost J_{50}, averaged over 20 simulations, for each policy type using a expected objective function. Fig. 2-4(a) shows results for a uniform disturbance realization, while Fig. 2-4(b) shows results for a vertex-only disturbance realization.

It is immediately clear that the predicted costs now provide a more accurate ap-



(a) Uniform Disturbance

(b) Vertex-Only Disturbance

Figure 2-4: Minimization of Expected Cost

proximation of the incurred cost, compared to the disturbance free case (Fig. 2-2). This is due to the additional disturbance-based terms in the objective function (2.46), which varies with the type of disturbance realization. The predicted cost does a reasonable job approximating the incurred cost for the uniform disturbance realization, though it still underestimates the incurred cost for the vertex-only disturbance realization.

As anticipated in Section 2.4.2, for those policies in which $\mathbf{M}$ is fixed, the expected incurred cost is the same as the disturbance free incurred cost (Fig. 2-2), and the increase in predicted cost is equal to the additional terms in (2.46). This is not case the case for AFP-BT and AFP-Full, which identify new solutions in order to decrease the total predicted cost. The predicted cost relationships for each disturbance realization differ from the disturbance free case only in that CT-Opt now achieves the second-lowest predicted cost.

There are two key indicators in the results of Fig. 2-4 which suggest that the expected objective function is capable of making more accurate predictions than the disturbance free objective function. First, note that the qualitative predicted cost ranking precisely matches the qualitative incurred cost ranking for both disturbance realizations. While this result clearly is not sufficient to make a similar claim for other problems, it does suggest that the expected objective is superior to the disturbance

free objective in predicting behavior for *this* system.

Second, observe that AFP-Full and AFP-BT utilize the expected objective function to significantly decrease their incurred costs. In particular, whereas AFP-Full previously had the highest incurred costs (Fig. 2-2), with the expected objective it now achieves the lowest incurred costs of the RMPC approaches considered. The simulation trajectories corroborate these findings (Fig. 2-5). Unlike the disturbance free cost (Fig. 2-3), the AFP approaches in Fig. 2-5 have additional information about the disturbance environment within their objective functions. Under this optimization, the AFP approaches select a plan which still minimizes control usage but also regulates the system much closer to the origin, reducing the input needed to react to large disturbances.

Worst Case Objective

Fig. 2-6 compares the predicted cost $J^*(x_0)$ and the incurred cost J_{50}, averaged over 20 simulations, for each policy type using a worst case objective function. Fig. 2-6(a) shows results for a uniform disturbance realization, while Fig. 2-6(b) shows results for a vertex-only disturbance realization.

As expected for a worst case metric, the predicted costs have increased significantly, and in many cases exceed the corresponding incurred costs. Note that because both disturbance realizations have the same worst case disturbance sequence, the predicted costs are again equal for each approach in Fig. 2-6(a) and Fig. 2-6(b).

Qualitatively, the predicted cost relationship for both disturbance realizations is the same as those for the expected cost. Again, AFP-Full has the lowest predicted cost, followed by CT-Opt, AFP-BT, and finally CT-Nil/AFP-Fix-Nil. However, the incurred cost relationship has changed significantly. The most obvious distinction is that AFP-Full, as with the disturbance free objective, has the highest incurred costs. Here, the margin between AFP-Full and the other approaches is significant. This clearly violates the DVH, and again suggests that in the general case, $J^*(x_0)$ may not be a reliable predictor of closed-loop performance.

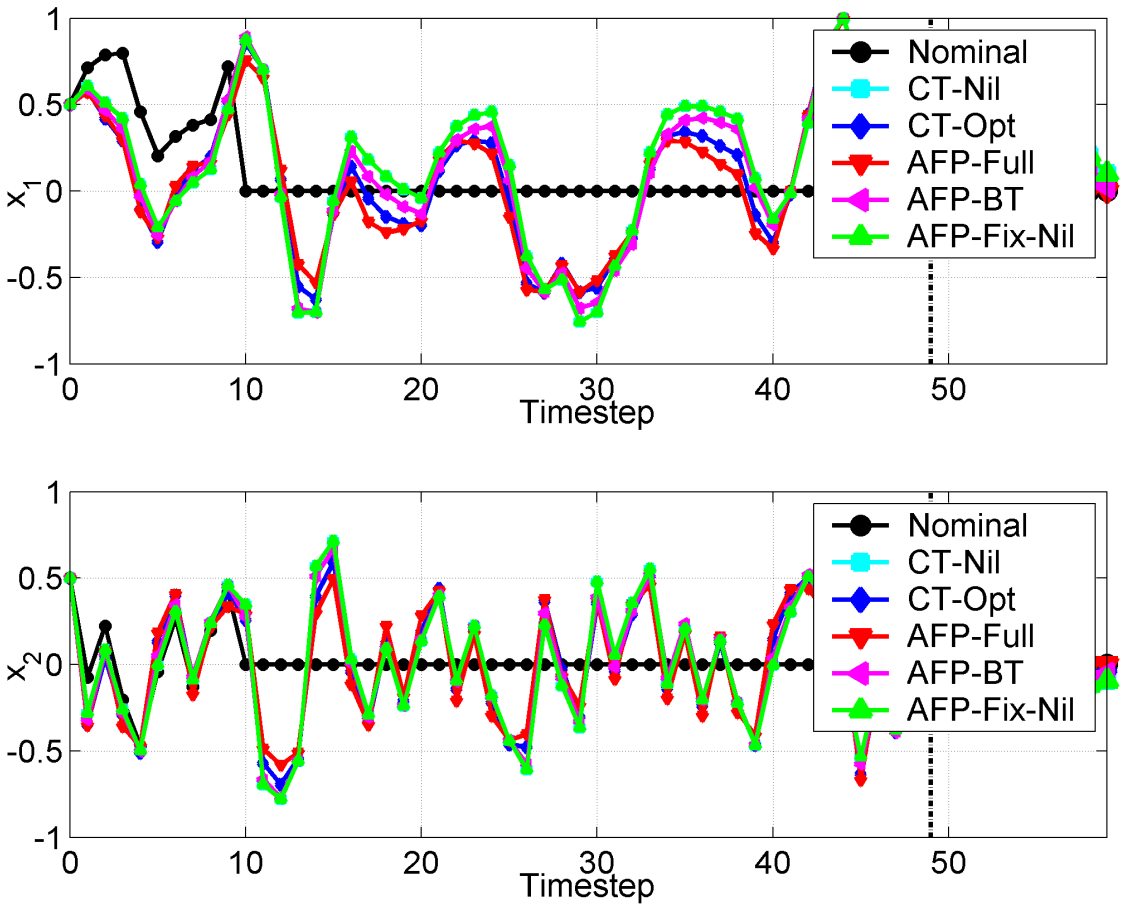
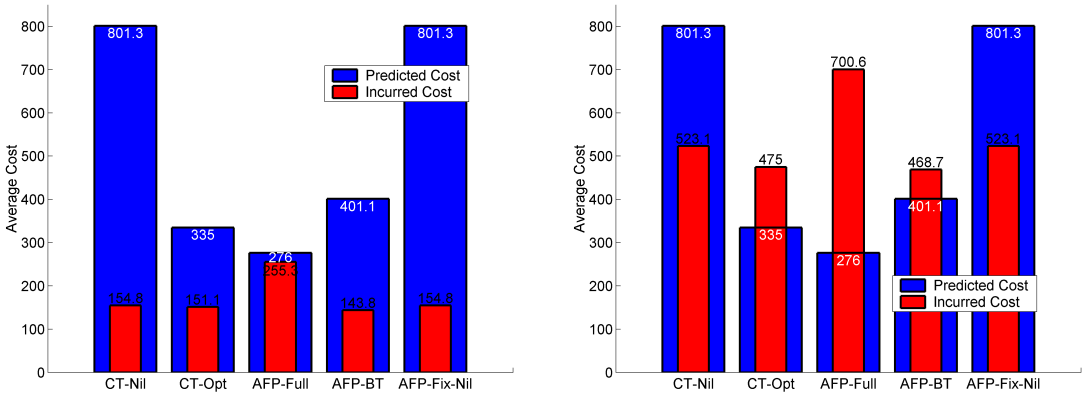


Figure 2-5: Sample state history and current state plan for each RMPC approach after 50 timesteps have been completed, using a expected objective and uniform disturbance realization. The nominal policy becomes infeasible at timestep 10.

2.5.4 Complexity

Each RMPC optimization can be represented as either a quadratic program or a semidefinite program, depending on the objective function used (Section 2.4). Thus the most reliable theoretical measures of an optimization's complexity are the number of decision variables and the number of inequality constraints. Table 2.2 compares the number of decision variables and inequality constraints for each policy using a quadratic objective function, for both the general case and the simulation example of Section 2.5.1. The additional constraints for the semidefinite program are equal in size for all approaches, and thus are not addressed here.

For the simulation example, $\mathbf{M}$ contributes only 45 additional decision variables



(a) Uniform Disturbance

(b) Vertex-Only Disturbance

Figure 2-6: Minimization of Worst Case Cost

to AFP-Full and 9 to AFP-BT. The remaining V_w decision variables are necessary, along with $N_w c_a$ constraints, to define the maximization in (2.37) for an infinity-norm-bounded disturbance set with linear constraints [28]. On the other hand, the AFP-BT-Fixed policy does not need these extra variables or constraints, since its maximization can be evaluated off-line. Note the dramatic increase in complexity by simply allowing $\mathbf{M}$ to be optimized on-line.

Fig. 2-7 compares the average simulation runtime over all 20 simulations for each feasible RMPC policy for the six cases considered in the previous section. All problems were solved on a 2.59-GHz computer with 512 MB of RAM.

Clearly, there is a runtime advantage for the policies of nominal complexity (CT-Nil, CT-OPT, and AFP-Fix-Nil) compared to the policies with more decision variables (AFP-Full and AFP-BT). For each problem considered in the previous section, the nominal-complexity policies are the fastest, followed by AFP-BT, then AFP-Full. This is consistent with all previous analysis and the number of degrees of freedom available for each optimization type.

The worst case objective function leads to imposing simulation runtimes, even for the nominal-complexity policies, because of the need to perform a semidefinite optimization. However, even the quadratic programming approaches can have long runtimes - note that the expected-cost runtimes for AFP-Full are similar to the worst

Table 2.2: Problem Complexity, Quadratic Objective

RMPC Type	**General # Vars.**	**General # Cons.**	**Sim # Vars.**	**Sim # Cons.**
CT-Nil	mN	c_a	10	44
CT-Opt	mN	c_a	10	48
AFP-Full	$mN + V_w +$ $mn_wN(N-1)/2$	$(N_w+1)c_a$	515	966
AFP-BT	$mN + V_w +$ $(N-1)mn_w$	$(N_w+1)c_a$	479	966
AFP-Fix-Nil	mN	c_a	10	46

$$
\begin{aligned}
c_a &= c_y N + c_t \\
V_w &= \begin{cases} c_a \text{ if } \mathbf{S}_w \text{ 1-norm-bounded} \\ n_w N c_a \text{ if } \mathbf{S}_w \text{ } \infty\text{-norm-bounded} \\ c_w N c_a \text{ if } \mathbf{S}_w \text{ polyhedral} \end{cases} \\
N_w &= \begin{cases} 2n_w N \text{ if } \mathbf{S}_w \text{ 1- or } \infty\text{-norm-bounded} \\ c_w N \text{ if } \mathbf{S}_w \text{ polyhedral} \end{cases}
\end{aligned}
$$

case-cost runtimes for AFP-BT. This seems to suggest that the expected cost results in a particularly hard problem for AFP-Full.

Fig. 2-7, when considered alongside Figs. 2-2, 2-4, and 2-6, represents visually the important tradeoff between complexity and conservativeness – though this tradeoff has been shown to be invalid for the incurred cost, in general.

2.6 Conclusion

This chapter investigated the relationship between CT and AFP policies for RMPC, through an extended theoretical and numerical analysis. It is clear that in general, AFP policies achieve lower predicted costs, while CT policies can be optimized with lower runtimes. However, while there are nice theoretical properties relating the CT and AFP paradigms, these properties do not accurately reflect closed-loop behavior using receding horizon control. A simple, conventional example has shown that despite significant increases in computational complexity, additional decision variables do not always improve, and sometimes degrade, the closed-loop performance of the RMPC policies. Of the objective functions considered for the example, the expected

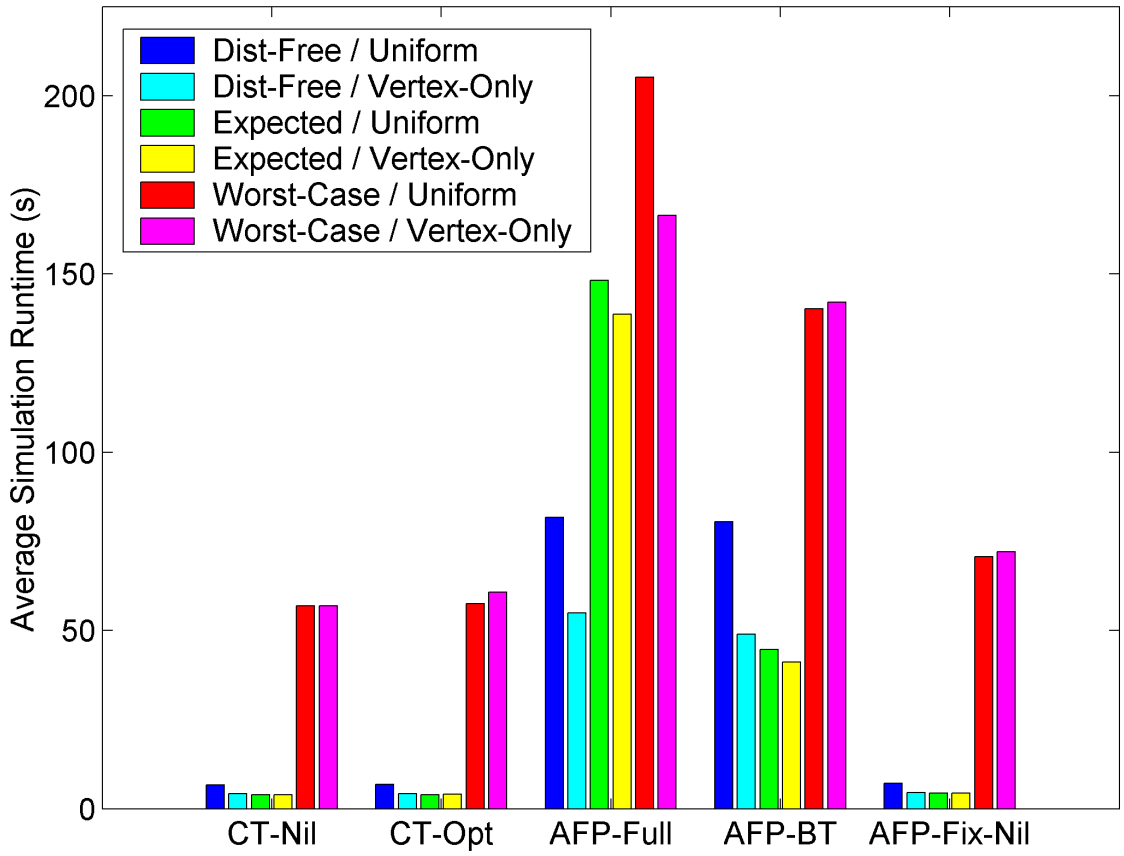


Figure 2-7: Average simulation runtime for each combination of objective function and disturbance realization considered above

objective appears most capable of predicting the actual incurred cost, and enables the AFP approaches to recover an incurred cost advantage relative to CT approaches. Nonetheless, deviations from the cost-to-go estimate have been shown to be possibly significant enough to influence the choice of a cost-minimizing controller, in general.

Appendix: Extending the Cost-to-Go

The objective of this appendix is to provide some preliminary analysis of how to better approximate the infinite-horizon cost in the RMPC optimization (2.6) using the cost-to-go function $f_N(\cdot)$.

Suppose that beyond the first N timesteps, the system only applies the linear terminal control law $\kappa(x) = Kx$ as its sole control strategy. Since this control law has been selected in CT and AFP to satisfy all constraints within the terminal set, where the system must arrive after N timesteps, the constraints can be effectively ignored beyond the horizon length N.

The objective here is to compute the cost incurred beyond timestep N, using the control law $u = Kx$ and ignoring all constraints. Of course, due to the persistent disturbances, the infinite-horizon cost will typically be infinite; instead, consider the finite-horizon cost

$$J = \sum_{t=0}^{M-1} x_{N+t}^T Q x_{N+t} + u_{N+t}^T R u_{N+t}, \tag{2.62}$$

with cost-to-go horizon length M; later this cost will be extended via $M \to \infty$.

Substituting $u = Kx$ into the system model (2.1) yields

$$x_{t+1} = A_{CL} x_t + G w_t, \tag{2.63}$$

where $A_{CL} = A + BK$. Similarly, substituting $u = Kx$ into the cost function (2.62) yields

$$J = \sum_{t=0}^{M-1} x_{N+t}^T Q_{CL} x_{N+t}, \tag{2.64}$$

where $Q_{CL} = Q + K^T R K$. Concatenating the state and disturbance as

$$\mathbf{x} = \begin{bmatrix} x_N^T & x_{N+1}^T & \cdots & x_{N+M-1}^T \end{bmatrix}^T, \tag{2.65}$$

$$\mathbf{w} = \begin{bmatrix} w_N^T & w_{N+1}^T & \cdots & w_{N+M-2}^T \end{bmatrix}^T, \tag{2.66}$$

respectively, then the cost can be written simply as

$$J = \mathbf{x}^T \mathbf{Q} \mathbf{x}, \tag{2.67}$$

where $\mathbf{Q} = I_M \otimes Q_{CL}$.

It is straightforward to show that

$$\mathbf{x} = \mathbf{A} x_N + \mathbf{G} \mathbf{w}, \tag{2.68}$$

where

$$\mathbf{A} = \begin{bmatrix} I_n \\ A_{CL} \\ A_{CL}^2 \\ \vdots \\ A_{CL}^{M-1} \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ G & 0 & \cdots & 0 \\ A_{CL}G & G & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A_{CL}^{M-2}G & A_{CL}^{M-3}G & \cdots & G \end{bmatrix}. \tag{2.69}$$

Substituting (2.68) into (2.67) yields

$$\begin{aligned} J &= (\mathbf{A} x_N + \mathbf{G} \mathbf{w})^T \mathbf{Q} (\mathbf{A} x_N + \mathbf{G} \mathbf{w}) \\ &= x_N^T \mathbf{A}^T \mathbf{Q} \mathbf{A} x_N + 2 x_N^T \mathbf{A}^T \mathbf{Q} \mathbf{G} \mathbf{w} + \mathbf{w}^T \mathbf{G}^T \mathbf{Q} \mathbf{G} \mathbf{w}. \end{aligned} \tag{2.70}$$

Next, this appendix considers how to apply this cost for each type of objective function.

Disturbance Free Cost

Assuming $\mathbf{w} \equiv 0$, (2.70) becomes

$$J_{df} = x_N^T \mathbf{A}^T \mathbf{Q} \mathbf{A} x_N. \tag{2.71}$$

Suppose that $P = \mathbf{A}^T\mathbf{Q}\mathbf{A}$. Using (2.69), note that

$$P = \sum_{t=0}^{M-1} (A_{CL}^t)^T Q_{CL} A_{CL}^t. \tag{2.72}$$

Now suppose $M \to \infty$, such that

$$P = \sum_{t=0}^{\infty} (A_{CL}^t)^T Q_{CL} A_{CL}^t; \tag{2.73}$$

this will be finite if A_{CL} is asymptotically stable. By multiplying and adding terms to both sides, the resulting relationship is

$$\begin{aligned} Q_{CL} + A_{CL}^T P A_{CL} &= Q_{CL} + A_{CL}^T \left(\sum_{t=0}^{\infty} (A_{CL}^t)^T Q_{CL} A_{CL}^t \right) A_{CL} \\ &= \sum_{t=0}^{\infty} (A_{CL}^t)^T Q_{CL} A_{CL}^t = P, \end{aligned}$$

or

$$Q + K^T R K + (A + BK)^T P (A + BK) = P. \tag{2.74}$$

But this is precisely the steady-state relationship (2.43) already being used to identify P. This validates the choice of P for the disturbance free objective, even taking post-horizon disturbances into account. Note that because the disturbances are assumed to be zero within the objective function, this infinite-horizon cost is actually finite.

Expected Cost

Take the expectation of (2.70), such that

$$J_{ex} = \mathbb{E}\left[x_N^T \mathbf{A}^T \mathbf{Q} \mathbf{A} x_N + 2 x_N^T \mathbf{A}^T \mathbf{Q} \mathbf{G} \mathbf{w} + \mathbf{w}^T \mathbf{G}^T \mathbf{Q} \mathbf{G} \mathbf{w} \right]. \tag{2.75}$$

The first term is independent of the disturbance and can be taken out of the objective. Because the disturbances are assumed to be independent and zero-mean, the second

term is equal to zero. Thus

$$J_{ex} \;=\; x_N^T P x_N + \mathrm{tr}\left[\mathbf{G}^T\mathbf{Q}\mathbf{G}\mathbf{C}_w\right], \tag{2.76}$$

where $\mathbf{C}_w = I_{M-1} \otimes C_w$ and $P = \mathbf{A}^T\mathbf{Q}\mathbf{A}$, as with the disturbance free cost.

In the current formulation of this objective function, only the first term of (2.76) is included as the cost-to-go. Yet the second term of (2.76) contains no decision variables, and thus has the nice property that it is *fixed off-line*. If this term is added to the objective function for each RMPC approach, a fixed quantity will be added to each predicted cost, but the incurred costs will remain the same.

As $M \to \infty$, the second term in (2.76) becomes infinite; is there a finite choice for M that makes sense? A smart choice for this setup would be to choose M such that $N + M$ equals the number of iterations of Algorithm 2.1, i.e. 50 in Section 2.5.

Worst Case Cost

Finally, consider the worst case cost problem, modeled as

$$\begin{aligned} J_{wc} \;=\; x_N^T P x_N + \max_{\mathbf{w}} \quad & 2x_N^T\mathbf{A}^T\mathbf{Q}\mathbf{G}\mathbf{w} + \mathbf{w}^T\mathbf{G}^T\mathbf{Q}\mathbf{G}\mathbf{w} \\ \text{s.t.} \quad & \mathbf{w} \in \mathbf{S}_w^{M-2}. \end{aligned} \tag{2.77}$$

Assuming $\mathbf{S}_w$ is polyhedral, the maximization is a convex optimization which is quadratic in $\mathbf{w}$. However, because of the presence of x_N, this optimization *cannot* be solved for a specific value off-line.

Chapter 3

Robust Trajectory Planning using Mixed-Integer Linear Programming

3.1 Introduction

This chapter proposes a trajectory planner which applies mixed-integer linear programming (MILP) to identify optimal, robust trajectories. Mixed-integer linear programming provides a very general framework for describing and solving problems which involve discrete decisions as well as continuous variables. Recent research has identified a variety of algorithms for effective MILP-based UAV trajectory planning; see Section 1.2.2 for a summary. When possible, the results in this chapter are presented for a generalized problem formulation, to emphasize the versatility of this approach.

The planner proposed in this chapter integrates and refines two recently-developed MILP planning algorithms. The first is the Robust Safe but Knowledgeable (RSBK) algorithm [10, 23, 50], which itself integrates notions of vehicle safety and an intelligent cost-to-go map with CT (Section 2.3.1) for robustness. The second algorithm is Variable MILP [51, 59], which includes several features designed for efficient on-line

MILP optimization in three-dimensional environments. When combined, these two algorithms represent many of the key innovations necessary for a UAV to effectively and autonomously plan trajectories in an uncertain environment.

This planner, named Efficient RSBK, also introduces several novel refinements to improve problem efficiency. Refinements to the 2-norm approximation, referred to here as variable-density constraint selection, strategically allocate constraints in order to reduce the total number of constraints without significantly impacting performance. A detection radius is modeled to represent a UAV's incomplete situational awareness. Additionally, the Selective CT technique, which allows CT to be performed on problems with varying timestep lengths, is proposed. Though this technique cannot guarantee a feasible optimization at every timestep, it does ensure robust constraint satisfaction at specified timesteps with a limited number of decision variables.

This chapter is structured as follows. Section 3.2 presents the problem statement, an extension of the problem posed in Section 2.2.2. Section 3.3 justifies the choice of CT for achieving robustness to disturbances, and poses a general form of the robust MILP trajectory planning problem. Section 3.4 briefly reviews the RSBK algorithm within the context of this formulation. Section 3.5 proposes a set of refinements to the RSBK approach; these refinements are either applications of Variable MILP [51] or are novel features proposed by this thesis. Several refinements require a more specific formulation of the vehicle model, and are introduced in Section 3.6. The effectiveness of this algorithm is demonstrated through a series of simulations in Section 3.7, followed by concluding remarks in Section 3.8.

3.2 Problem Statement

Consider the linear time-invariant dynamics with an additive disturbance and output constraints,

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t + Gw_t, && (3.1)\\ y_t &= Cx_t + Du_t \in \mathbf{S}_y, && (3.2)\\ w_t &\in \mathbf{S}_w, && (3.3) \end{aligned}$$

where $x_t \in \mathbb{R}^n$ is the state, $u_t \in \mathbb{R}^m$ is the input, $y_t \in \mathbb{R}^p$ is the output, and $w_t \in \mathbb{R}^{n_w}$ is an additive disturbance acting on the state. This disturbance is unknown at current and future times, but is known to lie within the polytopic set

$$\mathbf{S}_w = \{w \mid E_w w \leq f_w\}, \tag{3.4}$$

which contains the origin in its interior. It is assumed that (A, B) is stabilizable and that the full state is available at all timesteps. The timesteps $t \in \mathbb{N}_\infty$ are assumed to have the fixed duration δ. The system's objective is to achieve the goal state x_G.

The output is to remain constrained within the bounded but *non-convex* set $\mathbf{S}_y$; in Chap. 2, $\mathbf{S}_y$ was assumed to be polytopic and contain the origin in its interior. Without loss of generality, this set can be described as a convex environment $\mathbf{S}_{y|0}$ containing a set of convex *avoidance regions*. To render the problem tractable for a MILP optimization engine, assume that there are a finite number of avoidance regions, and that both the environmental and avoidance constraints are polyhedral. Suppose the i-th avoidance region is denoted by $\mathbf{S}_y^i$, and define the polyhedral constraints

$$\begin{aligned} \mathbf{S}_y^0 &= \{y \in \mathbb{R}^p \mid E_y^0 y \leq f_y^0\}, && (3.5)\\ \mathbf{S}_y^i &= \{y \in \mathbb{R}^p \mid E_y^i y \leq f_y^i\} \ \ \forall\, i \in \mathbb{N}_{1,N_{obs}}, && (3.6) \end{aligned}$$

where $f_y^0 \in \mathbb{R}^{n_0}$, $f_y^i \in \mathbb{R}^{n_i} \ \ \forall\, i \in \mathbb{N}_{1,N_{obs}}$, and N_{obs} is the number of obstacles/avoidance

regions. Then the output constraint (3.2) is equivalent to the relation

$$\begin{cases} y_{t+j|t} \in \mathbf{S}_y^0, \\ y_{t+j|t} \notin \mathbf{S}_y^i \quad \forall\ i \in \mathbb{N}_{1,N_{obs}}. \end{cases} . \tag{3.7}$$

This general form may include environmental bounds, physical obstacles, velocity and acceleration bounds, and many other types of constraints.

3.3 CT-Based Trajectory Planning Formulation

In this section, a general form of the robust MILP trajectory planning problem is posed. This formulation uses receding horizon control (RHC) over a fixed planning horizon of length N, and does not require a specific form of the vehicle dynamics. However, due to the presence of disturbances in (3.1), it is necessary to incorporate RMPC techniques to maintain robust feasibility.

Of the RMPC techniques analyzed in Chap. 2, Disturbance CT (Section 2.3.1) is selected here as the most viable approach for on-line path planning. Because the disturbance feedback policy $M_i^o\ \forall\ i \in \mathbb{N}_{1,N-1}$ is selected off-line, the tightened constraints can be computed off-line, preventing an increase in the complexity of the on-line optimization. In this manner, the decision space of the on-line optimization remains the same as nominal MPC, allowing for efficient real-time computation. Section 2.5.3 has indicated that in general, higher-complexity policies may not necessarily achieve a lower incurred cost, despite this significant increase in computational complexity. Perhaps most significantly, CT is the only approach considered which allows non-convex constraints, a necessary prerequisite for operation in obstacle fields.

Algorithm 3.1. RHC-MILP with CT

1: Compute CT tightening sets (3.13)-(3.14)
2: **for** $t = 0$ to $t = \infty$ **do**
3: Take measurement of current state: $x_{t|t} = x_t$
4: Solve optimization (3.8)-(3.12)
5: Apply first input: $u_t = u^*_{t|t}$
6: **end for**

The robust MILP trajectory planning problem can be written in the general form

$$\min_{u_{\cdot|t}} \quad J(x_t) = f_N(x_{t+N|t}) + \sum_{j=0}^{N-1} f(x_{t+j|t}, u_{t+j|t}) \tag{3.8}$$

$$\text{s.t.} \quad x_{t+j+1|t} = Ax_{t+j|t} + Bu_{t+j|t}, \tag{3.9}$$

$$y_{t+j|t} = Cx_{t+j|t} + Du_{t+j|t} \in \mathbf{S}_{y|j} \ \forall \ j \in \mathbb{N}_{N-1}, \tag{3.10}$$

$$x_{t+N|t} \in \mathbf{S}_{x|N}, \tag{3.11}$$

$$x_{t|t} = x_t, \tag{3.12}$$

where the CT tightened constraint sets $\mathbf{S}_{y|j}$, clarified below, may be non-convex, and $\mathbf{S}_{x|N}$ is subject to the criteria (2.22)-(2.24) for constraint tightening, repeated here:

$$\begin{aligned} x_{t+N|t} &\in \mathbf{S}_{x|N} = \mathbf{S}_t \ominus L_{N-1}\mathbf{S}_w, \\ Ax + B\kappa(x) + L_{N-1}w &\in \mathbf{S}_t \ \ \forall \ x \in \mathbf{S}_t, \ \ \forall \ w \in \mathbf{S}_w, \\ Cx + D\kappa(x) &\in \mathbf{S}_{y|N-1} \ \ \forall \ x \in \mathbf{S}_t. \end{aligned}$$

This optimization is performed repeatedly, in accordance with Algorithm 3.1.

Because the output constraints (3.10) are non-convex, special care must be taken in applying the tightening sequence (2.20). Since the environmental constraints $\mathbf{S}^0_y$ are convex, the original form of (2.20) can be used,

$$\begin{aligned} \mathbf{S}^0_{y|0} &= \mathbf{S}^0_y, \\ \mathbf{S}^0_{y|j+1} &= \mathbf{S}^0_{y|j} \ominus (CL_j + DM^o_{j+1})\mathbf{S}_w, \quad \forall \ j \in \mathbb{N}_{N-2}. \end{aligned} \tag{3.13}$$

The avoidance regions, however, must be *expanded*, instead of tightened, since they are

to be avoided. It is straightforward to show that this can be accomplished by simply replacing the Pontryagin difference operation with the Minkowski sum operator:

$$\begin{aligned} \mathbf{S}^i_{y|0} &= \mathbf{S}^i_y, \quad \forall\, i \in \mathbb{N}_{1,N_{obs}}, \\ \mathbf{S}^i_{y|j+1} &= \mathbf{S}^i_{y|j} \oplus (CL_j + DM^o_{j+1})\mathbf{S}_w, \quad \forall\, j \in \mathbb{N}_{N-2}, \quad \forall\, i \in \mathbb{N}_{1,N_{obs}}. \end{aligned} \tag{3.14}$$

Denote the tightened forms of the polyhedral environmental and obstacle constraints as

$$\mathbf{S}^0_{y|j} = \{y \in \mathbb{R}^p \mid E^0_{y|j} y \leq f^0_{y|j}\}, \quad \forall\, j \in \mathbb{N}_{N-2}, \tag{3.15}$$

$$\mathbf{S}^i_{y|j} = \{y \in \mathbb{R}^p \mid E^i_{y|j} y \leq f^i_{y|j}\}, \quad \forall\, j \in \mathbb{N}_{N-2}, \quad \forall\, i \in \mathbb{N}_{1,N_{obs}}. \tag{3.16}$$

For the system to satisfy the avoidance region constraints (3.16) for the ith obstacle, it is only necessary that at least one of the row constraints *not* be satisfied. This "OR" condition is effectively captured in the MILP formulation by using binary variables. In particular, the constraints for avoiding the *tightened* obstacle $\mathbf{S}^i_{y|j}$ can be written in the linear form

$$E^i_{y|j} y_{t+j|t} \geq f^t_{y|j} - M\mathbf{b}^i_j, \tag{3.17}$$

$$\sum_{l=1}^{n_i} b^i_{j_l} \leq n_i - 1, \tag{3.18}$$

where $\mathbf{b}^i_j = \begin{bmatrix} b^i_{j_1} & \cdots & b^i_{j_{n_i}} \end{bmatrix}^T$, the $b^i_{j_l}$ are scalar binary variables constrained to the set $\{0, 1\}$, and M is sufficiently large to ensure a constraint is satisfied if its corresponding binary variable is 1. If at least one of the row constraints for $\mathbf{S}^i_{y|j}$ is not satisfied, the binary variable in the corresponding row of (3.17) can be feasibly set to 0, ensuring satisfaction of (3.18).

In Ref. [23], the constraint set $\mathbf{S}_y$ is allowed to be non-convex, but the details of this implementation are left for a specific example involving velocity/acceleration bounds, obstacles, and a two-step nilpotent terminal control policy. None of the formulation in the following two sections requires a specific form of the vehicle dynamics.

	Algorithm 3.2. RSBK
1:	Compute CT tightening sets (3.13)-(3.14)
2:	Compute cost-to-go map
3:	**for** $t = 0$ to $t = \infty$ **do**
4:	Take measurement of current state: $x_{t\|t} = x_t$
5:	If environment has changed, perform lines 1-2
6:	Solve optimization (3.19),(3.9)-(3.12),(3.21)-(3.25)
7:	Apply first input: $u_t = u^*_{t\|t}$
8:	**end for**

3.4 Robust Safe but Knowledgeable (RSBK) Algorithm

The RSBK algorithm [10, 23, 50] integrates multiple recently-proposed techniques for MILP-based trajectory optimization, resulting in a planner which can react to several kinds of uncertainty while guaranteeing robust feasibility. The algorithm builds on existing research which incorporates CT-based robustness with MILP trajectory planning [8, 48, 49], and has been demonstrated for both state-based [50] and disturbance-based [23] feedback. The following sub-sections review two of the algorithm's other features, trajectory safety and cost-to-go mapping. The RSBK algorithm is given in Algorithm 3.2.

3.4.1 Trajectory Safety

A planner is said to achieve trajectory safety if any trajectory identified by the planner is guaranteed to be feasible at all future timesteps, regardless of the length of the planning horizon. Trajectory safety has been successfully achieved using MILP trajectory planning [9, 46, 47] by requiring the terminal state to fall within a basis state, defined as a state or sequence of states in which the vehicle can remain at all future timesteps without violating constraints. Common forms of the basis state include a zero-velocity position for full-stop vehicles (e.g., helicopters and quadrotors) and loiter circles for no-stop vehicles (e.g., airplanes) [47]. MILP-based trajectory safety may also be achieved by computing both an optimal solution and a feasible "rescue

path" at each planning interval [46].

There is a strong relationship between the basis state and the notions of invariant sets [60, 61] and robust invariant sets [57, 62] using in RMPC. This relationship has been exploited by algorithms using CT, including RSBK, to identify arbitrary basis states. Using CT, because the terminal state is already constrained to lie within the robust positively invariant set $\mathbf{S}_t$ defined by (2.23)-(2.24), any solution to the optimization (3.8)-(3.12) can be used to ensure trajectory safety. In particular, this is achieved by constructing the candidate solution (2.25)-(2.28), which is guaranteed to be robustly feasible at the following timestep. Thus, the terminal set acts as a basis state.

In the RSBK algorithm, the terminal set is allowed to be parameterized as a "decision variable," such that the objective (3.8) takes the form

$$\begin{aligned} \min_{u_{\cdot|t}, \mathbf{S}_{x|N}} \quad & J(x_t) \\ \text{s.t.} \quad & (3.9) - (3.12). \end{aligned} \tag{3.19}$$

While a maximal robust invariant set [63] ensures the largest feasible region, computation of such a set is often intractable in a non-convex environment. Alternatively, traditional forms of the basis state such as hover points and loiter circles can be used. Nonetheless, it remains a non-trivial task to ensure constraint satisfaction of these parameterized terminal sets in the presence of disturbances. One useful option is use a nilpotent control policy, such that $L_{N-1} = 0$; in this case, the set $\mathbf{S}_t$ need only be nominally invariant.

Note that trajectory safety implies that RSBK is an *anytime* algorithm, such that the actual optimization is not necessary for feasibility once a single feasible solution has been found. As long as the vehicle is operating in a portion of the environment in which it has perfect knowledge (Section 3.5.3), it is guaranteed to satisfy all constraints.

3.4.2 Cost-to-Go Map

The RSBK algorithm also incorporates an intelligent cost-to-go which uses the physical layout of the environment to estimate the cost of a full trajectory to the goal. The resulting cost map allows for efficient computation of short, detailed plans, while incorporating knowledge of future decisions. This idea was originally proposed in Ref. [42] for two-dimensional environments, then expanded in Refs. [44, 45] to include feasibility guarantees and Ref. [43] to incorporate a three-dimensional environment. Because of the robust feasibility and trajectory safety guarantees already included in RSBK, it is not necessary for the cost map to include a feasible trajectory to the goal; it should simply provide a reasonable approximation of the remaining cost. The Efficient RSBK algorithm constructs this cost map using the algorithm proposed in Refs. [23, 43] and summarized below.

The cost map assigns a cost estimate to a set of visibility graph nodes, including the start, goal, and certain points on the obstacle boundaries. For a 2-D environment, inclusion of the obstacle vertices is sufficient; however, for three-dimensional environments, the shortest path is more likely to pass obstacles on their edges, instead [64]. For this reason, it is appropriate to sample points along the edges to be included in the cost map calculation. In Ref. [43], a node is placed on each obstacle vertex at ground level ($z = 0$) and at each midpoint of edges above ground level ($z > 0$).

Once the visibility graph nodes have been selected, a linear program is used to identify mutually visible pairs of nodes, which are then connected via an edge [23]. Each edge is associated with a cost metric which estimates the actual cost incurred traversing the edge. Here, a simple weighted distance metric is used,

$$d(x_j, x_k) = r_d \|x_j - x_k\|_2, \tag{3.20}$$

where $r_d > 0$. A network optimization algorithm, such as Djikstra's algorithm, is then used to associate each node m with a minimum cost C_m to the goal from that node (Fig. 3-1).

The trajectory ultimately identified by the RSBK optimization includes three

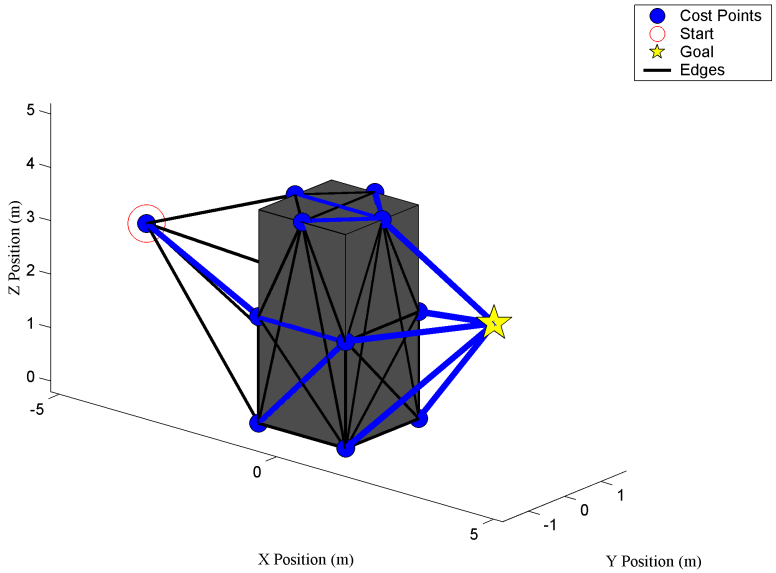


Figure 3-1: Sample cost-to-go map; minimum-cost paths are indicated by blue edges.

components: the trajectory over the planning horizon N, a path from the terminal state to a cost map node, and the shortest path on the visibility graph from that node to the goal (Fig. 3-2). The cost of the second portion is found using the distance metric (3.20), while the cost of the third portion is the cost C_m identified previously. By assigning a binary variable to each possible cost map node, the optimization can identify the cost-minimizing node on-line [42]. The resulting cost-to-go function and constraint set is

$$\begin{aligned} f_N(x_{t+j|t}) &= d(x_{t+j|t}, x_{vis}) + \sum_{m=1}^{N_{CP}} b_m^V C_m, && (3.21) \\ x_{vis} &= \sum_{m=1}^{N_{CP}} b_m^V x_m^V, && (3.22) \\ 1 &= \sum_{m=1}^{N_{CP}} b_m^V, && (3.23) \\ b_m^V &\in \{0,1\} \quad \forall\, m \in \mathbb{N}_{1,N_{CP}}, \end{aligned}$$

where x_{vis} is the selected cost point, N_{CP} is the total number of map nodes, and x_m^V

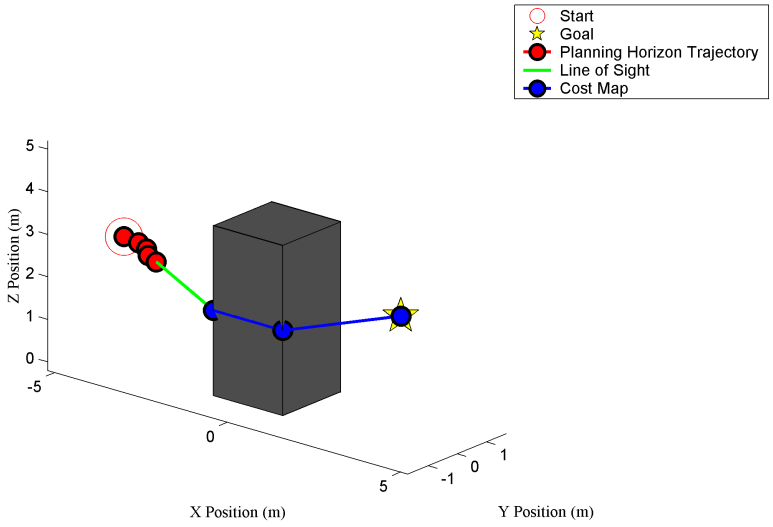


Figure 3-2: Sample RSBK Cost Map Plan

and C_m are the position and cost of the m-th map node, respectively. The weighted 2-norm $d(x_{t+j|t}, x_{vis})$ must be approximated using linear constraints; see Section 3.5.4.

Because the selected cost point x_{vis} is to be visible from the terminal state $x_{t+N|t}$, the line-of-sight path connecting them (Fig. 3-2) should be free of obstacles. This can be checked by enforcing the obstacle constraints at N_V^I interpolated sample points on the path, such that

$$E_x^i \left[\left(1 - \frac{e}{N_V^I + 1} \right) x_{t+N|t} + \frac{e}{N_V^I + 1} x_{vis} \right] \geq f_x^i - M \mathbf{b}_e^{Vi}, \tag{3.24}$$

$$(\forall\ e \in \mathbb{N}_{1,N_V^I},\quad \forall\ i \in \mathbb{N}_{1,N_{obs}^x}) \qquad \sum_{l=1}^{n_i} b_{e_l}^{Vi} \leq n_i - 1, \tag{3.25}$$

where $\mathbf{b}_e^{Vi} = \begin{bmatrix} b_{e_1}^{Vi} & \cdots & b_{e_{n_i}}^{Vi} \end{bmatrix}^T$ is a vector of binary variables, and N_{obs}^x indicates the upper bound of those avoidance regions which correspond to physical obstacles in the environment (and are assumed to come first in the list of avoidance regions).

Algorithm 3.3. Efficient RSBK

1: Compute CT tightening sets (3.13)-(3.14)
2: Compute cost-to-go map
3: **for** $t = 0$ to $t = \infty$ **do**
4: Take measurement of current state: $x_{t|t} = x_t$
5: If environment has changed, perform lines 1-2
6: Perform Algorithm 3.5
7: Solve optimization (3.19),(3.9)-(3.12),(3.21)-(3.25),(3.31)-(3.33)
8: Apply first input: $u_t = u^*_{t|t}$
9: **end for**

3.5 Refinements

The following two sections introduce the Efficient RSBK algorithm, an extension of the RSBK algorithm which incorporates elements of Variable MILP [51] and several novel refinements. Each of these refinements is discussed in turn below; Section 3.6 considers further extensions which require a more specific form of the system dynamics (3.1)-(3.2). The Efficient RSBK algorithm is given in Algorithm 3.3.

3.5.1 Selective CT

A key component of the Variable MILP algorithm [51] is the ability to vary the timestep lengths over the course of the plan. In this manner, a detailed plan can be developed in the short term, while a coarse plan is used in the far term (Fig. 3-3), where environmental knowledge is less certain and the plan waypoints are less likely to be used. However, this idea is difficult to apply to a CT-based algorithm such as RSBK, which requires the iterative modification of a candidate solution to guarantee robust feasibility [22]. If the timesteps are not evenly spaced, the state-space model (3.1) is generally *not* fixed, and the constraint-checked timesteps for one optimization may not align properly with those timesteps for the next optimization.

For the first, *optional* refinement, referred to as Selective CT, a state-space model corresponding to a fixed timestep length δ can be used to model variable timestep lengths by grouping together adjacent steps. To optimize a trajectory over a planning horizon of Δ seconds, $\lceil \Delta/\delta \rceil$ steps are necessary; however, the constraints $\mathbf{S}_{y|j}$ need

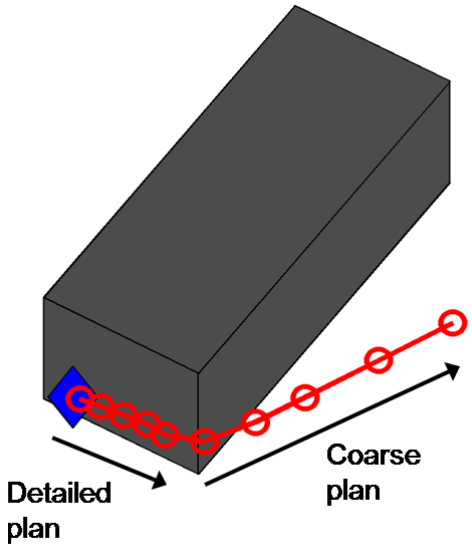


Figure 3-3: Sample trajectory plan constructed using Variable MILP or its robust analog, Efficient RSBK

not be enforced at every step. Instead, they can be applied only for those timesteps within the set N_C, which is assumed to contain the initial state $x_{t|t}$. Variable timesteps can then be modeled by requiring each input between elements of N_C to be equal to the element preceding it, i.e.

$$u_{t+j|t} = u_{t+j-1|t} \quad \forall\ j \notin N_C; \tag{3.26}$$

a similar approach is considered in Ref. [33]. In this case, the constraints (3.9)-(3.10) now take the form

$$x_{t+j+1|t} = Ax_{t+j|t} + Bu_{t+j|t}, \tag{3.27}$$

$$y_{t+j|t} = Cx_{t+j|t} + Du_{t+j|t} \in \mathbf{S}_{y|j}\ \forall\ j \in N_C, \tag{3.28}$$

One downside of this approach, and the reason Selective CT is considered an optional refinement, is that the optimization (3.19) cannot be guaranteed to be feasible at subsequent timesteps, due to the misalignment of constraint-checked timesteps between successive optimizations. Because of this, Algorithm 3.3 cannot be used with Selective CT. However, with an alternative form of the algorithm, Algorithm 3.4, the

Algorithm 3.4. Efficient RSBK with Selective CT

1: Compute CT tightening sets (3.13)-(3.14)
2: Compute cost-to-go map
3: **for** $t = 0$ to $t = \infty$ **do**
4: Take measurement of current state: $x_{t|t} = x_t$
5: Update candidate solution (2.25)-(2.28) using disturbance w_t
6: If environment has changed, perform lines 1-2
7: Perform Algorithm 3.5
8: Solve optimization (3.19),(3.26)-(3.28),(3.11)-(3.12), (3.21)-(3.25),(3.31)-(3.33)
9: **if** optimization is completed **do**
10: Apply first input: $u_t = u^*_{t|t}$
11: **else do**
12: Apply input from candidate solution: $u_t = \hat{u}_{t|t}$
13: **end if**
14: **end for**

system *can* be guaranteed to robustly satisfy all constraints at timesteps where they were initially checked and beyond the planning horizon N, even if the optimization itself is not always feasible.

Theorem 3.1 (Selective CT robust constraint satisfaction). *If the optimization in line 8 of Algorithm 3.4 is feasible at timestep $t = 0$, then a system operating using Algorithm 3.4 satisfies its constraints for all timesteps $j \in N_C$ or $j \geq N$.*

Proof. This proof builds off of the proof of robust feasibility for Disturbance CT (Theorem 2.3, Ref. [23]), referred to as the DCT proof below for clarity.

Suppose the optimization in line 8 of Algorithm 3.4 is feasible at timestep t. This means that the output constraint (3.28) is satisfied at timesteps $t + j, \quad \forall\ j \in N_C$, the terminal constraint (3.11) is satisfied at timestep $t + N$, and the initial constraint (3.12) is satisfied at timestep t. At the next timestep $t + 1$, construct the candidate solution (2.25)-(2.28) (line 5 of Algorithm 3.4).

Using the DCT proof as a guide, mane of these constraints can be proven to be satisfied by the candidate solution formed at timestep $t + 1$. Clearly, since the state constraint (3.8) is still being enforced at every timestep, the initial constraint (3.12) is satisfied at timestep $t + 1$. For the output constraint (3.28), the DCT proof indicates

the relationship

$$y_{t+j+1|t} \in \mathbf{S}_{y|j+1} \Rightarrow y_{t+j+1|t+1} \in \mathbf{S}_{y|j} \quad \forall\ j \in \mathbb{N}_{N-2}. \tag{3.29}$$

This implies that the output constraint (3.28) is *still* satisfied at timesteps $t+j,\ \forall\ j \in N_C/\{0\}$ - the same as in the previous optimization, minus the timestep already passed.

For the terminal constraint (3.11) and timestep $t+N$, the DCT proof indicates the relationship

$$x_{t+N|t} \in \mathbf{S}_{x|N} \Rightarrow \begin{cases} x_{t+N+1|t+1} \in \mathbf{S}_{x|N}, \\ y_{t+N|t+1} \in \mathbf{S}_{y|N-1}. \end{cases} \tag{3.30}$$

This implies not only that the terminal constraint (3.11) is satisfied at timestep $t+N+1$, but also that the output constraint (3.28) is satisfied at timestep $t+N$. In summary, if the output constraints are satisfied at timestep t for timesteps $t+j,\ \forall\ j \in N_C$, then they are also satisfied at timestep $t+1$ for timesteps $t+j,\ \forall\ j \in (N_C \bigcup \{N\})/\{0\}$.

Suppose the candidate solution (2.25)-(2.28) continues to be updated at subsequent timesteps, and that system inputs are applied from the candidate solution alone (line 12 of Algorithm 3.4). Then, given that the optimization in line 8 of Algorithm 3.4 is feasible at timestep t, the output constraints will be satisfied for timesteps $t+j,\ \forall\ j \in N_C$ or $j \geq N$. The proof is completed by simply setting $t = 0$. $\diamondsuit$

Note that the constraint tightening sets (3.13)-(3.14) are still computed for all timesteps, since (1) subsequent tightening steps, including those at timesteps in N_C, depend on all previous tightening steps, and (2) intermediate tightening steps are actually used in the construction of the candidate solution.

Since Theorem 3.1 does *not* guarantee that optimizations at future timesteps will be feasible, there is no obvious guarantee that Algorithm 3.4 can successfully move beyond the first feasible optimized trajectory plan at $t = 0$. However, in practice, this approach does achieve comparable performance to Algorithm 3.3, where constraints are enforced at every timestep; see Section 3.7.

Algorithm 3.4 modifies Algorithm 3.3 by not only including (3.26)-(3.28), but by

allowing the system to default to the input generated by the candidate solution if the optimization is infeasible or takes too long to complete. This is the form of the optimization actually implemented (Section 3.7.1). In subsequent sections, the N_C terminology is used regardless of whether Selective CT is applied; if it is not, assume $N_C = \{0, 1, \ldots, N-1\}$.

3.5.2 Linear Interpolation Points

Another refinement used from Variable MILP [51] is the notion of linear interpolation points, placed between timesteps (here, $j \in N_C$) to check for obstacle avoidance. For most trajectory planning problems, it is necessary to extend the obstacle bounds to prevent the vehicle from "cutting corners" between discrete timesteps (Section 3.6.2). However, because this expansion distance is proportional to the timestep spacing, it may become overly restrictive, and possibly remove feasible trajectories, during coarse portions of the trajectory plan. Linear interpolation points provide a means for reducing the size of this expansion distance without adding additional state variables to the problem formulation.

To incorporate linear interpolation points in the Efficient RSBK formulation, the additional constraints

$$E_x^i \left[\left(1 - \frac{e}{N_j^I + 1} \right) x_{t+N_{C|j-1}} + \frac{e}{N_j^I + 1} x_{t+N_{C|j}} \right] \geq f_x^i - M \mathbf{b}_e^{ji}, \quad (3.31)$$

$$(\forall\ j \in N_C \backslash \{0\},\ \ \forall\ e \in \mathbb{N}_{1,N_j^I},\ \ \forall\ i \in N_{obs}^x) \quad \sum_{l=1}^{n_i} b_{e_l}^{ji} \leq n_i - 1, \quad (3.32)$$

are added, where $\mathbf{b}_e^{ji} = \begin{bmatrix} b_{e_1}^{ji} & \cdots & b_{e_{n_i}}^{ji} \end{bmatrix}^T$ is a vector of binary variables, $N_{C|j}$ indicates the jth timestep contained in N_C, and N_j^I is the number of interpolation points placed between timesteps $N_{C|j-1}$ and $N_{C|j}$. Better trajectory approximations may be achieved through non-linear interpolation [51], though it is not considered further here.

3.5.3 Detection Radius

If the UAV does not have perfect knowledge of its environment, it has been assumed (Section 1.3.1) that it does have perfect knowledge within some detection radius $R_d > 0$. If the vehicle discovers previously-unseen obstacles, then the previous solution to the MILP optimization may become infeasible, jeopardizing long-term feasibility and the anytime algorithm status bestowed by trajectory safety (Section 3.4.1). For this reason, it is necessary for the planning horizon trajectory to remain fully within the region of perfect knowledge.

For simplicity, assume that the detection region is implemented as an ∞-norm bound centered at the initial state x_t, such that the region of perfect knowledge for a particular optimization is given by the "detection box"

$$\mathbf{S}_{\text{det}} = \{x \mid \|x - x_t\|_\infty \leq R_d\}. \tag{3.33}$$

Because this constraint is convex and polyhedral, it can easily be added to the set of system constraints by combining it with the environmental constraints $\mathbf{S}_y^0$.

If the vehicle's situational awareness is restricted to the detection box, obstacles are added to the RSBK optimization only if some portion of the obstacle enters this detection box, indicating that the obstacle has been detected by the UAV. When this occurs, the cost-to-go map needs to be recomputed to account for these new obstacles (line 5 of Algorithms 3.3-3.4).

3.5.4 Variable-Density Constraint Selection

One of the critical challenges in using linear programming to model trajectory planning problems is finding an appropriate approximation for 2-norms. The 2-norm may be used to represent constraints such as bounds on absolute velocity and acceleration, distances between points, radar sites, and other constraints. Using many constraints, a 2-norm can be modeled quite accurately with linear programming, but at the expense of increased computational complexity. Conversely, a 2-norm modeled with few constraints may result in the trajectory favoring the (inaccurate) corners of the coarse

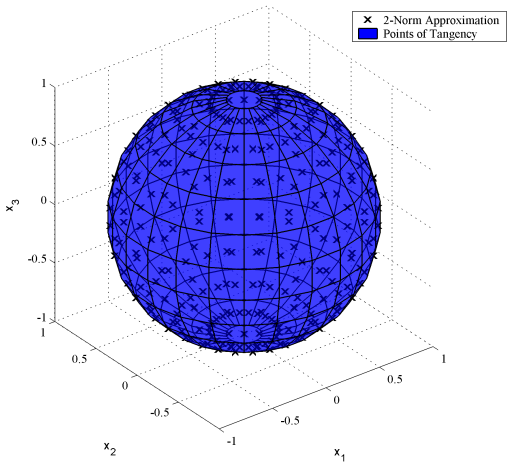


Figure 3-4: Approximating the 2-norm via spherical geometry-based sampling; 288 constraints are used in this figure.

2-norm approximation, leading to suboptimal trajectories. This section introduces a novel approach for utilizing this tradeoff between complexity and optimality, referred to here as variable-density constraint selection (VDCS).

Suppose that points in three-dimensional space are represented by the ordered triple $\mathbf{x} = (x, y, z)$. A typical 2-norm constraint can be represented by

$$\|\mathbf{x}\|_2 \leq R \quad \Rightarrow \quad x^2 + y^2 + z^2 \leq R^2, \tag{3.34}$$

where R is a positive scalar. This particular constraint is centered at the origin; the extension to non-zero origins is straightforward. One simple way to approximate this 2-norm with linear constraints is to sample planes tangent to points on the boundary of this sphere. At the point $\mathbf{p} = (p_x, p_y, p_z)$, where $p_x^2 + p_y^2 + p_z^2 = R^2$, the equation of the tangent plane inequality is simply

$$p_x x + p_y y + p_z z \leq R. \tag{3.35}$$

While this approach does create an outer approximation of the 2-norm, this over-approximation can be minimized by sampling a large number of well-spaced points around the sphere.

A traditional technique for approximating the 2-norm samples constraints using a spherical geometry representation,

$$\cos\left(\frac{2\pi i}{D}\right)\sin\left(\frac{2\pi j}{D}\right)x + \sin\left(\frac{2\pi i}{D}\right)\sin\left(\frac{2\pi j}{D}\right)y + \cos\left(\frac{2\pi j}{D}\right)z \leq R \qquad (3.36)$$
$$\forall\ i \in \mathbb{N}_{1,D/2},\quad \forall\ j \in \mathbb{N}_{1,D},$$

where the discretization level D is a positive, even constant. This approach uses $D^2/2$ constraints, and converges to a perfect sphere as $D \to \infty$.

While an accurate 2-norm is often necessary to achieve useful, smooth trajectories, the number of constraints needed to model the full sphere can quickly become excessive. In many cases, one or more 2-norm approximations may be necessary at every constraint-checked timestep. However, a detailed representation of the 2-norm is typically not necessary in every direction. For example, consider a 2-norm constraint on the velocity magnitude (Section 3.7), for a vehicle moving towards some goal waypoint. The optimized velocity is most likely to fall on the portion of the 2-norm approximation aligned with the linear segment connecting the vehicle with the goal. Conversely, constraints on the opposite side of the 2-norm approximation are extremely unlikely to be enforced, and need not be as closely spaced.

Using this logic, the Efficient RSBK algorithm applies a variable-density 2-norm approximation scheme, VDCS, which couples a coarse 2-norm approximation with additional regions of closely-packed constraints aligned with "directions of interest." These high-density regions are characterized by four parameters:

- the *direction of interest*, $\mathbf{d} = (d_x, d_y, d_z)$, assumed to be a unit vector;
- the *number of constraint rings* surrounding this vector on the 2-norm approximation, N_R;
- the *angular spacing* of the constraint rings, θ_R; and
- the *discretization level* of each constraint ring, D_R.

The approach begins with a low-D 2-norm approximation using spherical geometry, as in Fig. 3-5(a). Additional constraints are built around each direction of interest by

Algorithm 3.5. VDCS

Inputs: D, $\mathbf{d}$, N_R, θ_R, D_R

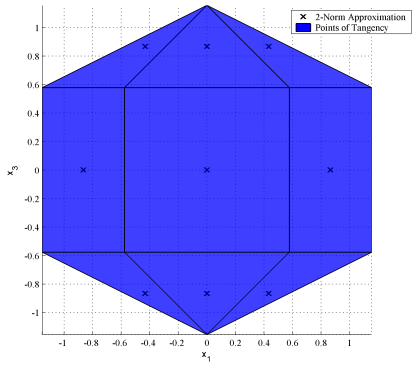
P: set of sampled 2-norm points/constraints

1: $P \leftarrow$ (3.36)
2: $P \leftarrow \mathbf{d}$
3: **for** $i = 1$ **to** $i = N_R$ **do**
4: $\quad \mathbf{r} \leftarrow$ random unit vector in $\mathbb{R}^3$
5: $\quad \mathbf{r} \leftarrow (\mathbf{d} \times \mathbf{r})/|\mathbf{d} \times \mathbf{r}|$
6: $\quad \mathbf{r} \leftarrow \mathbf{d}$ rotated about $\mathbf{r}$ by angle $\theta_R i$
7: $\quad$ **for** $j = 1$ **to** $j = D_R$ **do**
8: $\quad\quad P \leftarrow \mathbf{r}$
9: $\quad\quad \mathbf{r} \leftarrow \mathbf{r}$ rotated about $\mathbf{d}$ by angle $2\pi/D_R$
10: $\quad$ **end for**
11: **end for**

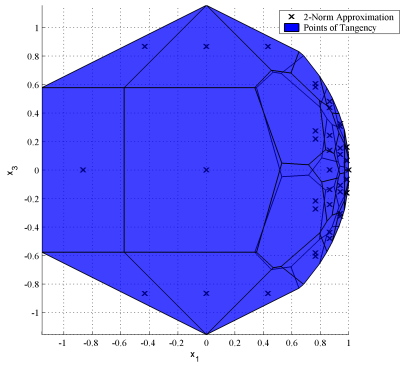
constructing N_R "constraint rings," spaced θ_R radians apart, and sampling equally-spaced constraints from each ring with the discretization level D_R.

The VDCS construction process is summarized in Algorithm 3.5, which assumes a single direction of interest $\mathbf{d}$. All rotations (lines 6 and 9) are performed using quaternions. Note line 4, in which $\mathbf{p}$ is initialized to a random unit vector; this direction is chosen randomly to minimize undesirable symmetry in the resulting 2-norm approximation. The algorithm builds the 2-norm approximation using $D^2/2 + N_R D_R + 1$ constraints: $D^2/2$ for the coarse approximation, $N_R D_R$ for the constraint rings, and 1 for the direction $\mathbf{d}$ itself.

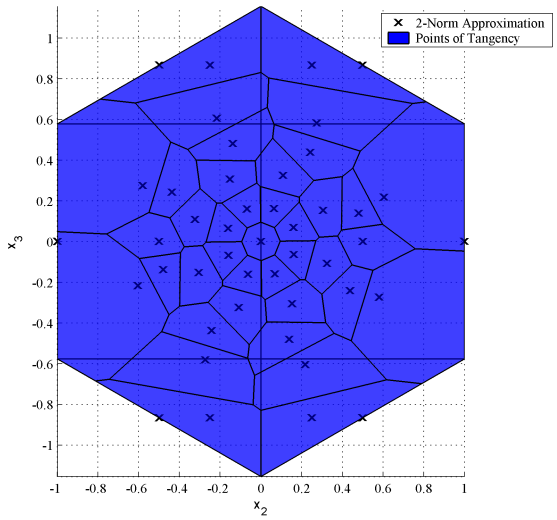
To demonstrate this technique, consider a 2-norm approximation with $D = 6$, $\mathbf{d} = (1, 0, 0)$, $N_R = 4$, $\theta_R = \pi/18$, and $D_R = 8$. Fig. 3-5(a) shows a side view of the 2-norm approximation if *only* the coarse representation using (3.36) is considered. Figs. 3-5(b) – 3-5(c), on the other hand, show two orthogonal views of the 2-norm approximation with the regional constraints also added. As shown, this hybrid approach successfully integrates regions of widely-spaced and closely-spaced constraints. Additionally, by using the same discretization level D_R for each constraint ring, the constraint density naturally increases moving towards $\mathbf{d}$ (Fig. 3-5(c)). Note that only 50 constraints have been used to construct this approximation, compared to 288 in Fig. 3-4.



(a) Coarse constraints only (side view)



(b) Coarse and regional constraints (side view)



(c) Coarse and regional constraints (front view)

Figure 3-5: Demonstration of coarse/region 2-norm approximation, with $D = 6$, $\mathbf{d} = (1, 0, 0)$, $N_R = 4$, $\theta_R = \pi/18$, and $D_R = 8$

The regional constraint parameters are easily adjusted to customize the nature of the regional constraints for the problem being considered. By increasing D_R from 8 to 16, a better regional 2-norm approximation can be achieved for a region of fixed size (Fig. 3-6(a)). By increasing θ_R from $\pi/18$ to $\pi/9$, the same number of constraints can be used to better approximate a larger region of the 2-norm (Fig. 3-6(b)). In fact, if N_R and θ_R are chosen such that $N_R\theta_R = 2\pi$, the entire circle can be sampled using constraint rings. Fig. 3-6(c) demonstrates this by increasing N_R from 4 to 18; note the overall asymmetry of the 2-norm approximation.

3.6 Model-Specific Refinements

Several additional refinements from Variable MILP are included in the Efficient RSBK framework which require a specific form of the vehicle state and/or dynamics. After these are provided, the additional refinements, obstacle expansion and obstacle reachable horizon, are discussed below. Note that although these refinements are discussed in the context of a single vehicle type, it is straightforward to apply these concepts to other vehicles.

3.6.1 Vehicle Dynamics

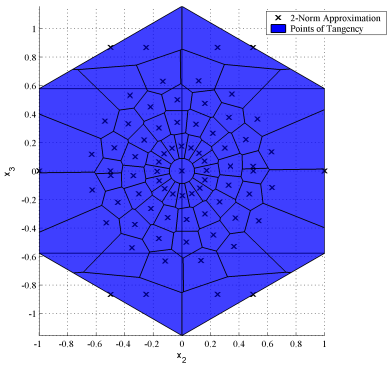
Consider a simple hovering vehicle with double integrator dynamics, often a reasonable approximation of a helicopter or quadrotor (Section 4.1). The vehicle state x and input u are broken down into the components

$$x = \begin{bmatrix} \mathbf{p}^T & \mathbf{v}^T \end{bmatrix}^T, \qquad u = \mathbf{a}, \tag{3.37}$$

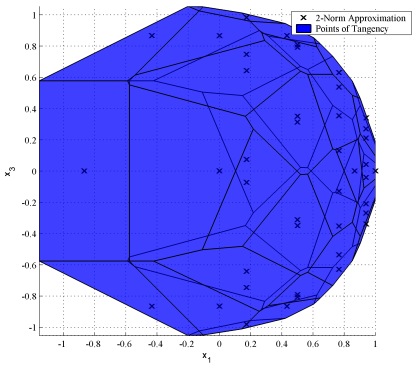
$$\mathbf{p} = \begin{bmatrix} p_x & p_y & p_z \end{bmatrix}^T, \qquad \mathbf{a} = \begin{bmatrix} a_x & a_y & a_z \end{bmatrix}^T, \tag{3.38}$$

$$\mathbf{v} = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix}^T, \tag{3.39}$$

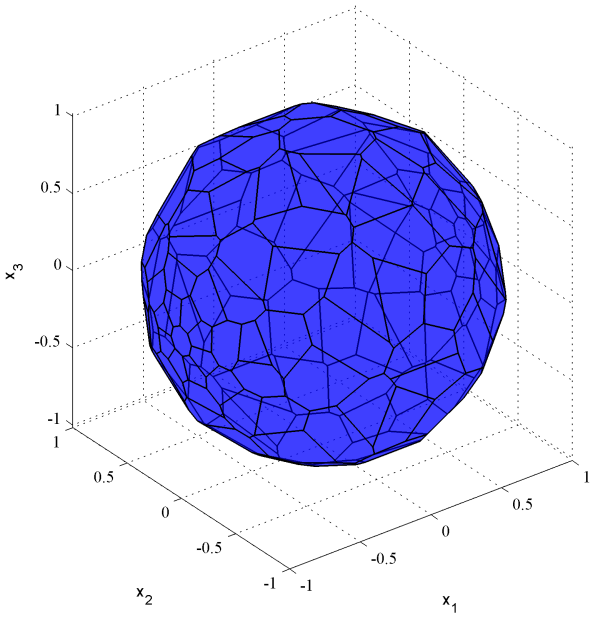
where $\mathbf{p}$, $\mathbf{v}$, and $\mathbf{a}$ are the position, velocity, and acceleration vectors, respectively, and each subscript indicates the coordinate direction. The state-space model (3.1)-(3.2)



(a) $D_R : 8 \rightarrow 16$ (front view)



(b) $\theta_R : \pi/18 \rightarrow \pi/9$ (side view)



(c) $N_R : 4 \rightarrow 18$ (entire sphere)

Figure 3-6: Effects of adjusting the regional constraint parameters.

takes the form

$$x_{t+1} = \begin{bmatrix} I_3 & \delta I_3 \\ 0_3 & I_3 \end{bmatrix} x_t + \begin{bmatrix} \frac{1}{2}\delta^2 I_3 \\ \delta I_3 \end{bmatrix} u_t + \begin{bmatrix} \frac{1}{2}\delta^2 I_3 \\ \delta I_3 \end{bmatrix} w_t, \tag{3.40}$$

$$y_t = \begin{bmatrix} I_3 & 0_3 \\ 0_3 & I_3 \\ 0_3 & 0_3 \end{bmatrix} x_t + \begin{bmatrix} 0_3 \\ 0_3 \\ I_3 \end{bmatrix} u_t; \tag{3.41}$$

note that this is simply the extension of (2.51) to three dimensions. If the output constraints $\mathbf{S}_y$ are decoupled between the state and input, (3.41) allows CT tightening to be decoupled as well. In this case, suppose $\mathbf{S}_x$ and $\mathbf{S}_u$ represent the decoupled state and input constraints, respectively; then (2.20) can be rewritten as

$$\begin{aligned} \mathbf{S}_{x|0} &= \mathbf{S}_x, \\ \mathbf{S}_{x|i+1} &= \mathbf{S}_{x|i} \ominus L_i \mathbf{S}_w, \quad \forall\, i \in \mathbb{N}_{N-2}, \\ \mathbf{S}_{u|0} &= \mathbf{S}_u, \\ \mathbf{S}_{u|i+1} &= \mathbf{S}_{u|i} \ominus M^o_{i+1} \mathbf{S}_w, \quad \forall\, i \in \mathbb{N}_{N-2}, \end{aligned}$$

If this decoupling is appropriate for the problem setup, it can dramatically decrease the complexity of the CT tightening implementation.

The vehicle has a maximum and minimum velocity and acceleration constraint in each coordinate direction, resulting in a set of convex, rectangular constraints. The environment is also assumed to be rectangular, with a maximum position in each coordinate direction. This set of constraints can be represented as

$$E^0_y = \begin{bmatrix} I_3 & 0_3 & 0_3 \\ 0_3 & I_3 & 0_3 \\ 0_3 & 0_3 & I_3 \\ -I_3 & 0_3 & 0_3 \\ 0_3 & -I_3 & 0_3 \\ 0_3 & 0_3 & -I_3 \end{bmatrix}, \quad f^0_y = \begin{bmatrix} \mathbf{p}_{\max} \\ \mathbf{v}_{\max} \\ \mathbf{a}_{\max} \\ -\mathbf{p}_{\min} \\ -\mathbf{v}_{\min} \\ -\mathbf{a}_{\min} \end{bmatrix}. \tag{3.42}$$

The velocity and acceleration are also subject to magnitude constraints

$$\|\mathbf{v}\|_2 \leq V_{\max}, \quad \|\mathbf{a}\|_2 \leq A_{\max}. \tag{3.43}$$

Finally, the environment contains a set of physical obstacles $\mathbf{S}_y^i \ \ \forall\ i \in \mathbb{N}_{1,N_{obs}}$, which are to be avoided.

3.6.2 Obstacle Expansion

In order to ensure that the UAV using Efficient RSBK does not violate any constraints at any point on its trajectory, constraints must be tightened according to three specific factors. All constraints must be tightened using the CT algorithm (Section 3.2) to achieve robustness to the additive disturbances in (3.40). Any position constraint must be tightened to account for the vehicle size, such that the vehicle can be represented as a point mass operating within a configuration space. Finally, any non-convex obstacles in the environment must also be expanded to prevent "corner-cutting," in which the trajectory plan is feasible at two consecutive discrete timesteps but intersects an obstacle between them.

There are multiple ways to prevent corner-cutting from taking place; the time- and velocity-based representation from Variable MILP [51] is used here. Each obstacle at optimization step j is expanded by the distance

$$d_{\text{safe}|j} = \delta_j^I V_{U|j} \frac{\sqrt{2}}{4}. \tag{3.44}$$

Here δ_j^I is the time interval between optimization step j and the previous step in which the obstacle constraints were checked, and can be reduced significantly by the use of linear interpolation points (Section 3.5.2). The quantity $V_{U|j}$ represents the maximum *speed* attainable by the vehicle at optimization step j, based on the initial velocity and the maximum/minimum acceleration bounds at each timestep.

3.6.3 Obstacle Reachable Horizon

Due to the constraints on the vehicle's position and velocity in (3.42)-(3.43), the UAV can only reach points within some finite region at a given optimization step. Just as each timestep is identified with a maximum attainable speed for (3.44), each timestep can also be associated with a maximum position horizon. This expanding rectangular box, proposed as part of Variable MILP [51], provides an outer bound on the set of positions the vehicle can reach at any given timestep.

A key advantage of this representation is that only those obstacles which intersect this box after the expansions in Section 3.6.2 need to be included in the MILP formulation [51]. Because those obstacles fully outside the box are guaranteed to be avoided, they will not affect the optimization solution, and can be removed from the optimization completely. This approach removes variables and constraints associated with these obstacles, potentially resulting in a significant runtime savings. Note, however, that all obstacles must still be included for those portions of the trajectory beyond the planning horizon, including the line-of-sight and cost map trajectories (Fig. 3-2).

If a detection radius is applied, it can essentially be treated as an additional reachable horizon, see Section 3.5.3.

3.7 Simulation Results

In this section, the effectiveness of the Efficient RSBK algorithm is demonstrated for several simulation examples.

3.7.1 Implementation

The Efficient RSBK algorithm is implemented in the form of Algorithm 3.4. The algorithm is written entirely in Java [65], and is solved using ILOG CPLEX 9.0 [66]. All optimizations were performed on a 2.00-GHz computer with 1 GB of RAM.

Each MILP optimization is constructed using the "warm start" approach [51],

wherein all components which are fixed for every optimization are added first and maintained through the planning process. For example, using VDCS, the coarse 2-norm using D is typically fixed during the warm start, while the direction of interest - and its related regional constraints - are unique to each optimization. In this manner, only essential operations are performed during each algorithm iteration, minimizing the time necessary to find a solution.

Simulated UAV behavior is based on the low-level controller introduced in Section 4.3, which sends vehicle state data to the MILP planner at 2 Hz. The simulated UAV is based on (3.40); controller gains have been selected to simulate realistic quadrotor behavior (Section 4.1).

3.7.2 Results

Consider the vehicle model from Section 3.6.1. The disturbances are bounded by the set $\mathbf{S}_w$ of (3.4), where

$$E_w = \begin{bmatrix} I_3 \\ -I_3 \end{bmatrix}, \quad f_w = \begin{bmatrix} \mathbf{w}_{\max}{}^T & -\mathbf{w}_{\min}{}^T \end{bmatrix}^T, \quad \mathbf{w}_{\max} = \mathbf{w}_{\min} = \gamma \mathbf{1}_3,$$

and $\gamma = 0.06$ is the disturbance level. A nilpotent control policy $M_j^o \ \forall \ j \in \mathbb{N}_{1,N-1}$ is applied, such that the terminal basis state for trajectory safety (Section 3.4.1) is simply a hover point, i.e.

$$\mathbf{v}_{t+N|t} = \mathbf{0}_3.$$

For simplicity, all obstacles are represented as three-dimensional boxes; the generalization to arbitrary obstacles is straightforward and does not introduce any theoretical complications.

The trajectory planner uses the objective function

$$f(\cdot) = r_p P_{t+j|t} + r_v V_{t+j|t} + r_a A_{t+j|t} + r_z b_j^z. \tag{3.45}$$

Here $P_{t+j|t}$, $V_{t+j|t}$, and $A_{t+j|t}$ represent 2-norm upper bounds on the position, velocity, and acceleration, respectively,

$$\|\mathbf{p}_{t+j|t} - \mathbf{p}_G\|_2 \leq P_{t+j|t}, \quad \|\mathbf{v}_{t+j|t}\|_2 \leq V_{t+j|t}, \quad \|\mathbf{a}_{t+j|t}\|_2 \leq A_{t+j|t}, \tag{3.46}$$

where $\mathbf{p}_G$ is the goal position; these 2-norms are approximated as specified in Section 3.5.4 in order to pose them as linear constraints. The discretization levels D are 12 for the position, 20 for the velocity, 16 for the acceleration, and 12 for the cost-to-go distance norm; for now, VDCS is not used. The binary b_j^z represents an optional altitude penalty constraint at each timestep,

$$h_{\max} \geq z_{t+j|t} - Mb_j^z, \tag{3.47}$$

where $h_{\max}$ is the desired altitude ceiling. Finally, $r_p = 1$, $r_v = 0.1$, $r_a = 0.1$, and $r_z = 800$ are positive scalar weights which can be adjusted by the operator. (In the cost-to-go distance norm, $r_d = 1$.)

The vehicle's size, modeled off of a Draganflyer quadrotor (Section 4.1), is $80 \times 80 \times 15$ cm^3. Most of the remaining simulation parameters introduced throughout this chapter are stated below:

$$\begin{aligned}
\mathbf{p}_{\min} &= \begin{bmatrix} -6 & -6 & 0 \end{bmatrix}^T, & \mathbf{p}_{\max} &= \begin{bmatrix} 6 & 6 & 6 \end{bmatrix}^T, \\
\mathbf{v}_{\min} &= \begin{bmatrix} -0.5 & -0.5 & -0.5 \end{bmatrix}^T, & \mathbf{v}_{\max} &= \begin{bmatrix} 0.5 & 0.5 & 0.5 \end{bmatrix}^T, \\
\mathbf{a}_{\min} &= \begin{bmatrix} -0.5 & -0.5 & -0.5 \end{bmatrix}^T, & \mathbf{a}_{\max} &= \begin{bmatrix} 0.5 & 0.5 & 0.5 \end{bmatrix}^T, \\
V_{\max} &= 0.5, & A_{\max} &= 0.5, \\
\delta &= 1, & R_d &= 2.5, \\
N_V^I &= 4.
\end{aligned}$$

Ten separate simulation scenarios are presented below. The properties of each simulation, including average solve time, number of constraints, and number of variables, are summarized in Table 3.1.

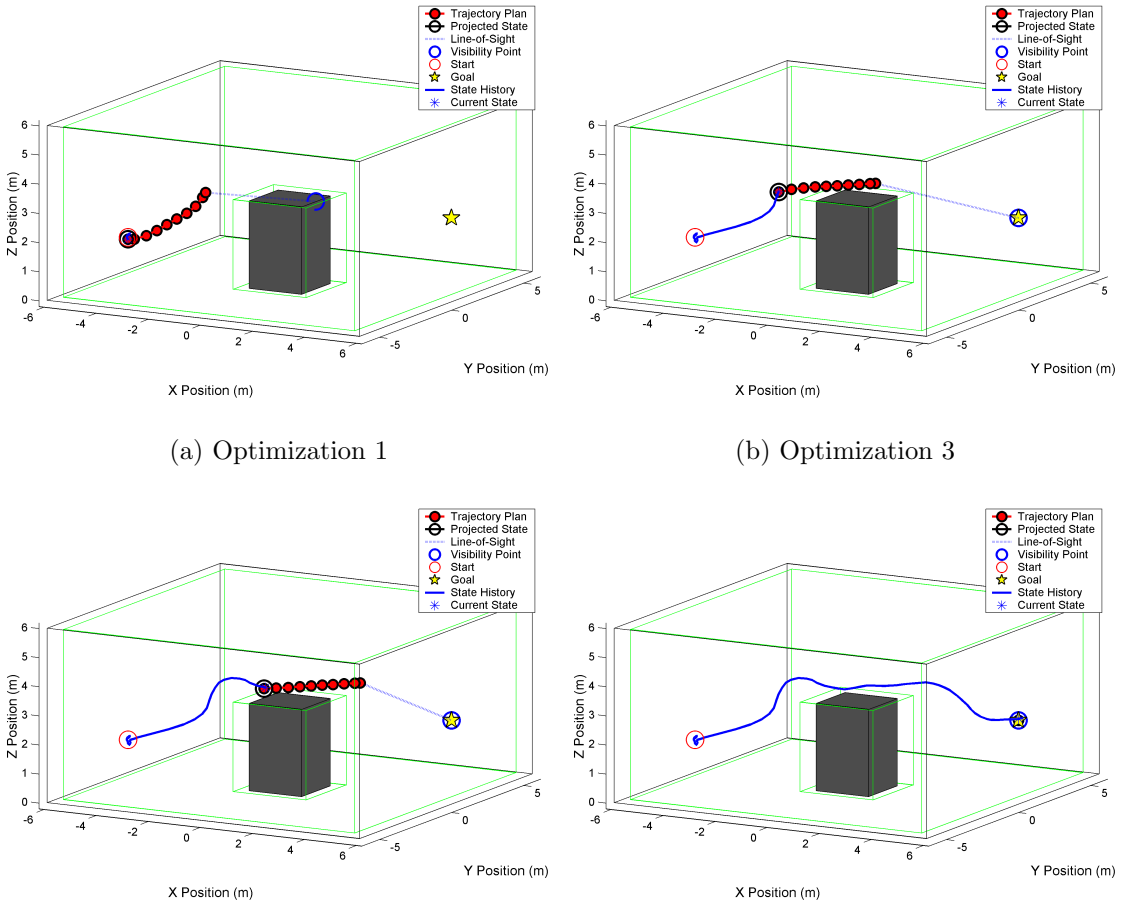
Table 3.1: Efficient RSBK Solution Properties

Figure Ref.	N	**Select. CT?**	**Alt. Pen.?**	**VDCS?**	**Det. Rad.?**	**Avg. Solve Time (ms)**	**Mean # Cons.**	**Mean # Vars.**
3-7	9	No	No	No	No	969	3851	205
3-8	9	No	Yes	No	No	723	3858	213
3-9	16	No	No	No	No	1672	6866	330
3-10	16	No	Yes	No	No	1423	6881	345
3-11	16	Yes	No	No	No	1041	3960	308
3-12	16	Yes	Yes	No	No	729	3970	318
3-13(a)	16	Yes	No	Yes	No	893	2914	308
3-13(b)	16	Yes	Yes	Yes	No	650	2922	317
3-14	16	Yes	No	Yes	Yes	406	2951	293
3-15	16	Yes	Yes	Yes	Yes	433	2959	301

Fig. 3-7 shows the state history and trajectory plan at four specific timesteps for a simulated vehicle using the Efficient RSBK algorithm to navigate to a goal waypoint, obstructed by an obstacle. In this scenario, no altitude penalty is enforced, and perfect knowledge of the environment is available to the vehicle. Furthermore, Selective CT is not applied: $N_C = \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$. The vehicle identifies a path directly over the obstacle to the goal, as indicated by the optimal trajectory plan at each timestep shown. The resulting smooth trajectory, shown in its entirety in Fig. 3-7(d), satisfies all hard constraints at every timestep, despite the disturbances present within the simulation.

Despite using a relatively large planning horizon, the first optimization (Fig. 3-7(a)) has such small timesteps that it is unable to see the goal x_G from its terminal state $x_{t+N|t}$. Instead, the terminal state is connected via line-of-sight to a cost map node located on top of the obstacle. Since the cost-to-go in this case is the sum of both this line-of-sight path and the cost map value, the optimization next seeks to position itself where it can see the goal directly. This can be observed in Fig. 3-7(b): the vehicle ascends at a higher rate than necessary to avoid the obstacle, so that the goal can be seen from the terminal state. Once this has been achieved, the trajectory reorients itself to follow the shortest path to the goal (Fig. 3-7(c)). Nonetheless, a "bump" remains in the overall trajectory (Fig. 3-7(d)).

Table 3.1 indicates an average optimization solve time for this scenario of 969 ms, which is slightly shorter than the timestep length of 1 second. However, some



(a) Optimization 1

(b) Optimization 3

(c) Optimization 5

(d) Optimization 13

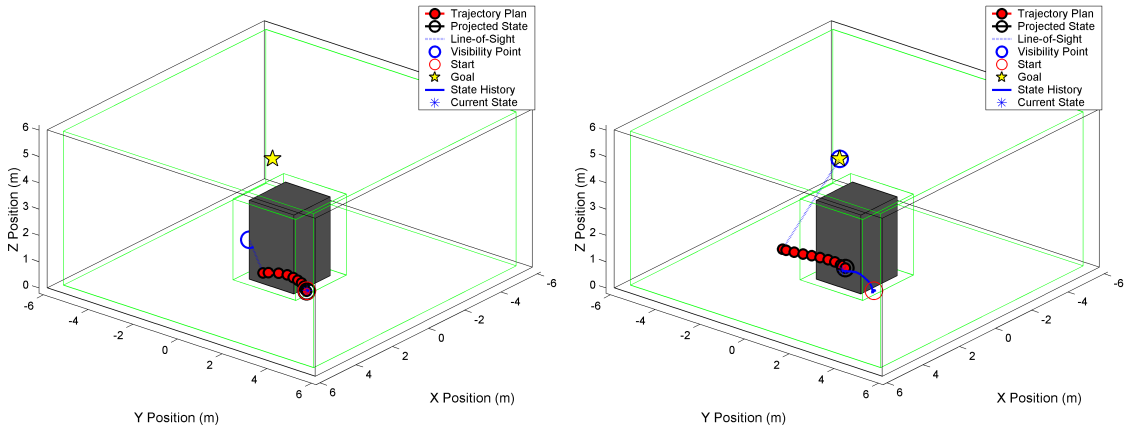
Figure 3-7: Flying over an obstacle using Efficient RSBK without Selective CT. Green lines indicate the configuration space.

of the initial optimizations do take longer than this timestep length. When this is the case, the vehicle is required to proceed beyond the first waypoint before receiving an updated trajectory plan, per line 12 of Algorithm 3.4. Since Efficient RSBK is an anytime algorithm, however, long-term feasibility is maintained regardless of the optimization duration.

Fig. 3-8 shows a scenario in which the waypoints of Fig. 3-7 are reversed - the start and goal are interchanged - but an altitude penalty is enforced at a height of 3 meters. Since the obstacle is also 3 meters high, this forces the trajectory planner to select a path around the obstacle, rather than over it, in order to minimize the trajectory cost. As in Fig. 3-7, the resulting trajectory (Fig. 3-8(d)) is smooth and satisfies all hard *and* soft constraints, including the altitude penalty. Additionally, no churning takes place: once the trajectory planner has committed to follow a path to the left, subsequent optimizations choose the same direction. However, the first optimization (Fig. 3-8(a)) is again unable to see the goal from the terminal state. A bump can be observed in its trajectory (Fig. 3-8(c)) where the vehicle cuts to the left in order to see the goal from its terminal state quickly.

Otherwise, the scenarios of Figs. 3-7 and 3-8 are relatively similar: the mean number of variables and constraints differ by less than 10 (Table 3.1), corresponding to the altitude penalty constraints and related binary variables. The average solution time is actually 25% smaller in this case than without the altitude penalty, despite the additional variables and constraints. This may be attributable to a reduction in the number of "directions" the planner can pursue. While the first scenario (Fig. 3-7) gives the trajectory planner significant freedom to select an optimal path, the second scenario (Fig. 3-8) restricts this freedom to a choice between moving left and moving right.

Figs. 3-9 and 3-10 show the state history and trajectory plan at four specific timesteps for the same scenarios as Figs. 3-7 and 3-8, respectively, but with the planning horizon N increased from 9 to 16. As expected, the first optimization for each scenario (Figs. 3-9(a) and 3-10(a)) is now able to see the goal position x_G from the terminal state without significant deviations from the shortest path.



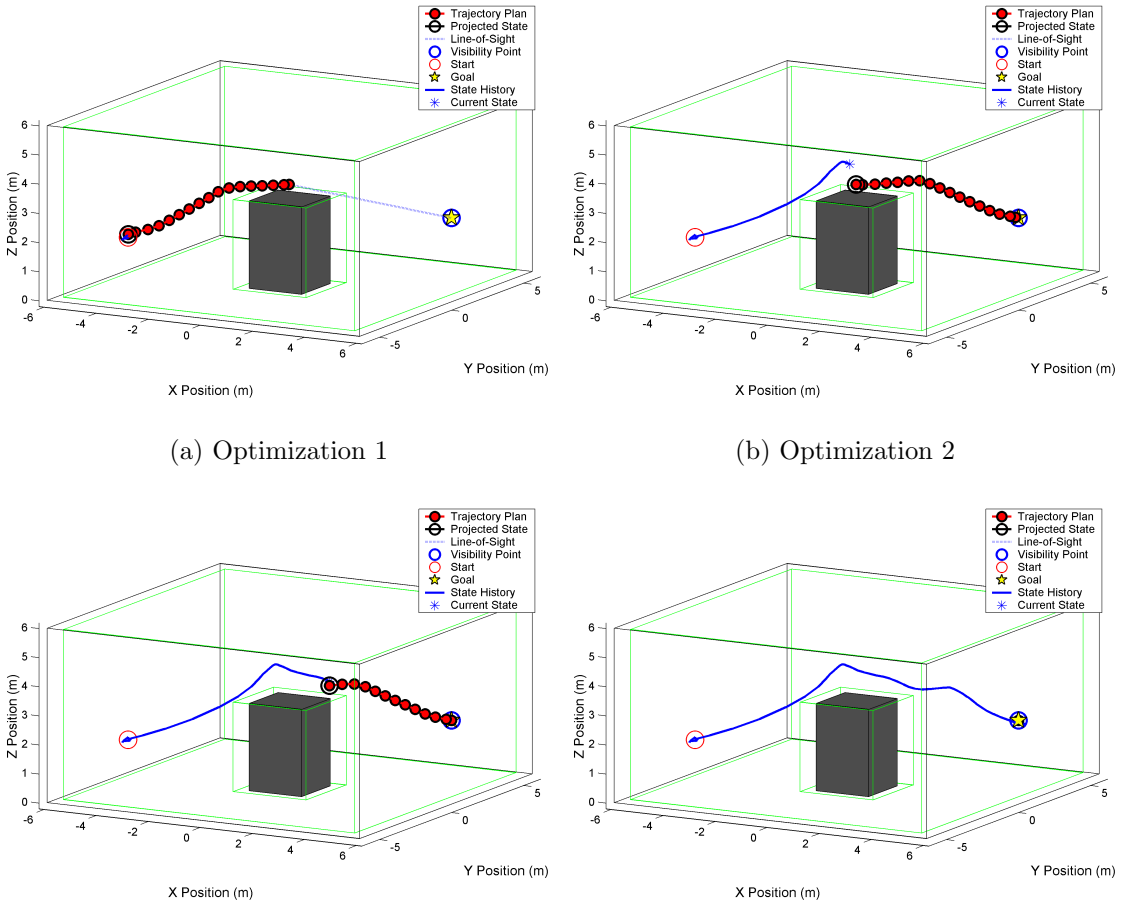
(a) Optimization 1

(b) Optimization 3

(c) Optimization 5

(d) Optimization 14

Figure 3-8: Flying around an obstacle using Efficient RSBK without Selective CT. Green lines indicate the configuration space.



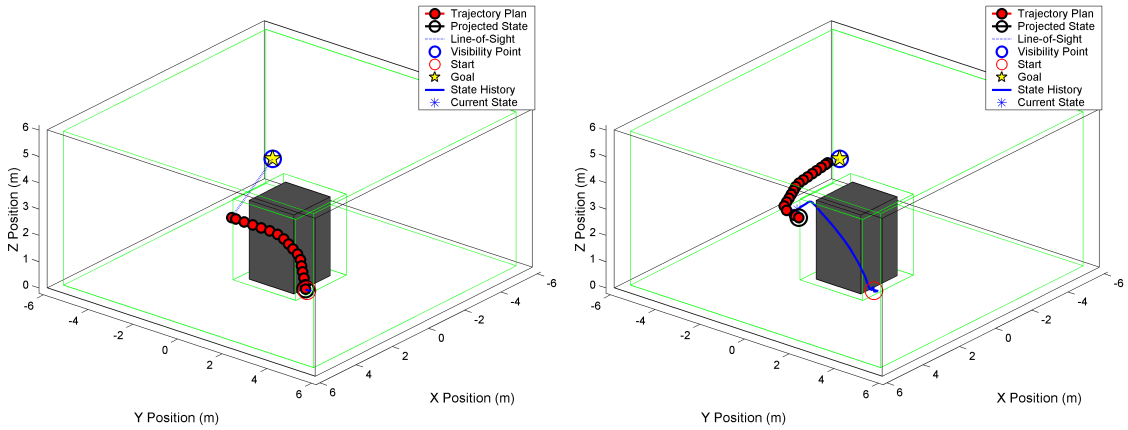
(a) Optimization 1

(b) Optimization 2

(c) Optimization 5

(d) Optimization 11

Figure 3-9: Flying over an obstacle using Efficient RSBK without Selective CT and a long planning horizon.



(a) Optimization 1

(b) Optimization 2

(c) Optimization 7

(d) Optimization 12

Figure 3-10: Flying around an obstacle using Efficient RSBK without Selective CT and a long planning horizon.

However, the additional degrees of freedom afforded by the extended horizon length dramatically increase the complexity of the optimization. As noted in Table 3.1 for each scenario, the average solution time and number of constraints have nearly doubled compared to $N = 9$, while the number of variables has increased by over 50%. In particular, the variance in the optimization runtimes has increased significantly: the initial optimizations for each scenario may take much less than a second or several seconds to complete.

These large variations in solution time can pose significant problems for the optimization engine, which has to measure the state of the vehicle (line 4 of Algorithm 3.4) *before* the optimization is performed (step 8 of Algorithm 3.4). To compensate, the implementation used here propagates the measured state forward based on a moving-average estimate of how long the optimization might take. If successive optimizations have significantly different solve times, there may be large discrepancies between the predicted and actual vehicle state once an optimization has terminated.

As Figs. 3-9 – 3-10 indicate, these disagreements may be significant enough to result in sub-optimal or even infeasible trajectories. For both scenarios, the first optimization solve time is much larger than the second, causing the second optimization to significantly overestimate the vehicle's progress. In this case, the vehicle may have to act quickly and erratically to catch up. In Fig. 3-9(b), all constraints are satisfied, but the trajectory has an unnecessary vertical deviation as the vehicle passes over the obstacle. In Fig. 3-10(b), the lateral deviation is significant enough to cause the UAV to violate the obstacle constraints.

It is thus clear that unnecessarily complex scenarios can lead to complications which undermine Efficient RSBK's guarantee of robust feasibility for a realistic implementation. The next four simulations show the significant improvements in performance achieved by introducing Selective CT and VDCS, each of which removes constraints and variables from the optimization.

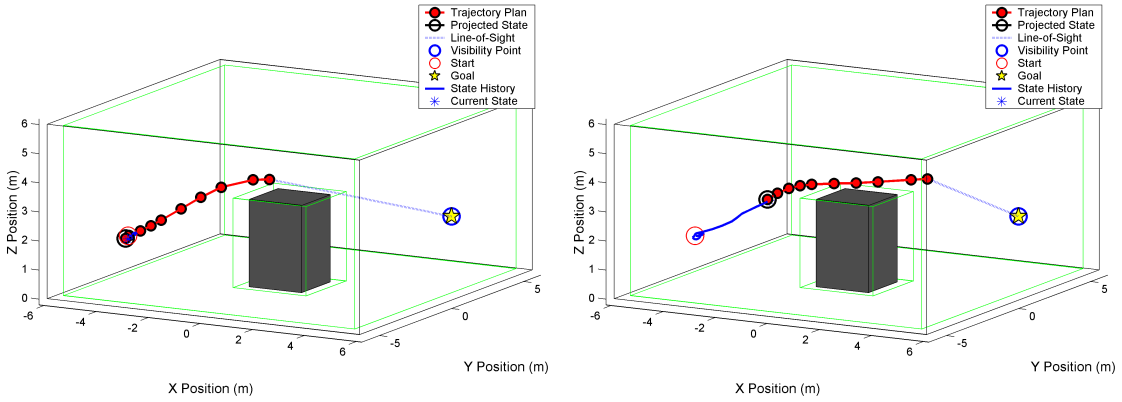
Figs. 3-11 and 3-12 show the resulting trajectory plans for the same scenarios as Figs. 3-9 and 3-10, respectively, but with Selective CT allowed; in particular, $N_C = \{0, 1, 2, 3, 4, 6, 8, 10, 13\}$. Linear interpolation points are used at the seven opti-

mization steps where the constraints are no longer being checked. By using Selective CT, the output constraints are only checked at 9 timesteps, but the planning horizon is still sufficiently long for the terminal state $x_{t+N|t}$ to see the goal x_G during the first optimization (Fig. 3-11(a)). In Figs. 3-11(a) – 3-11(b), it can be observed that the final two waypoints in each trajectory plan are closer in proximity than their predecessors, despite having the longest timestep length in the plan (3 seconds). This is due to the enforcement of trajectory safety, which requires a deceleration in the final optimization steps in order for the final state to be a zero-velocity basis state.

Even though Theorem 3.1 cannot guarantee robust feasibility for the MILP optimization, in practice it remains feasible for these scenarios and the simulation results which follow. In fact, compared to Figs. 3-9 – 3-10, the resulting trajectories are significantly smoother, and robustly satisfy all constraints. This is largely due to the significant decrease in problem complexity which results by not enforcing constraints at seven of the optimization steps (Table 3.1). In fact, the complexity of these scenarios is only incrementally larger than Figs. 3-7 – 3-8, which check constraints at the same number of optimization steps but have a much shorter total planning horizon N. Because the problem complexity has decreased so significantly, the solution times for successive optimizations have become more consistent, allowing the optimization engine to better predict and plan for the state's future location.

Figs. 3-13(a) and 3-13(b) show the final trajectories resulting from the same scenarios as Figs. 3-11 and 3-12, respectively, but with the original velocity 2-norm approximation replaced with the VDCS scheme of Section 3.5.4. In the first two scenarios, the velocity magnitude constraint is represented as a standard 2-norm approximation with $D = 20$, corresponding to 200 constraints at each timestep. For VDCS in Fig. 3-13, $D = 6$, $\mathbf{d} = \mathbf{p} - \mathbf{p}_G$, $N_R = 4$, $\theta_R = \pi/9$, and $D_R = 8$. These parameters correspond to the 2-norm approximation shown in Fig. 3-6(b), oriented in the direction from the vehicle to the goal for each optimization.

As seen in Fig. 3-13, the resulting trajectories have not degraded appreciably in quality compared to Figs. 3-11 – 3-12; both are relatively smooth and satisfy all constraints. The trajectory in Fig. 3-12(d) avoids the obstacle to the right,



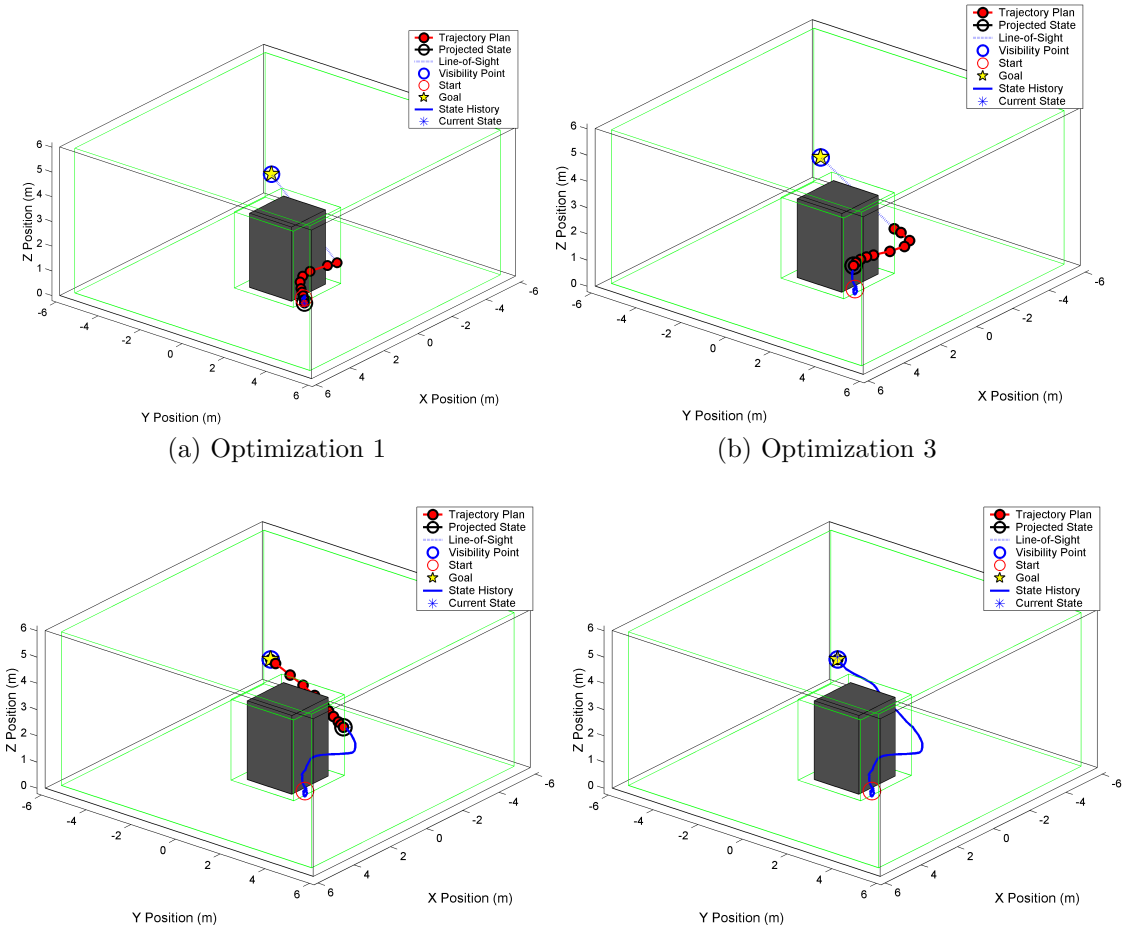
(a) Optimization 1

(b) Optimization 3

(c) Optimization 8

(d) Optimization 14

Figure 3-11: Flying over an obstacle using Efficient RSBK with Selective CT.



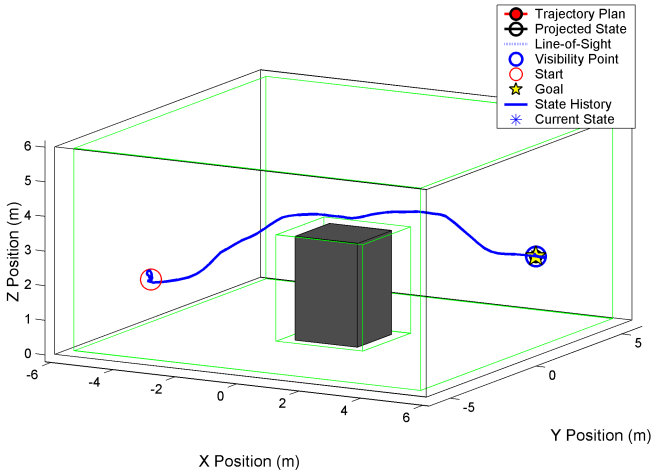
(a) Optimization 1

(b) Optimization 3

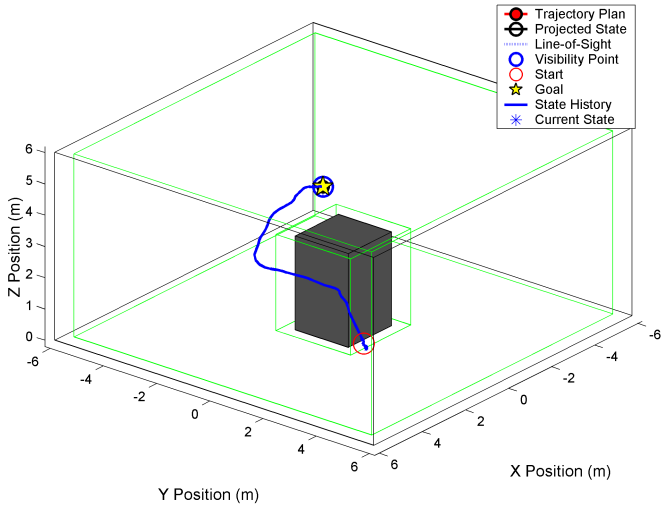
(c) Optimization 8

(d) Optimization 14

Figure 3-12: Flying around an obstacle using Efficient RSBK with Selective CT.



(a) Flying over an obstacle, Optimization 14



(b) Flying around an obstacle, Optimization 14

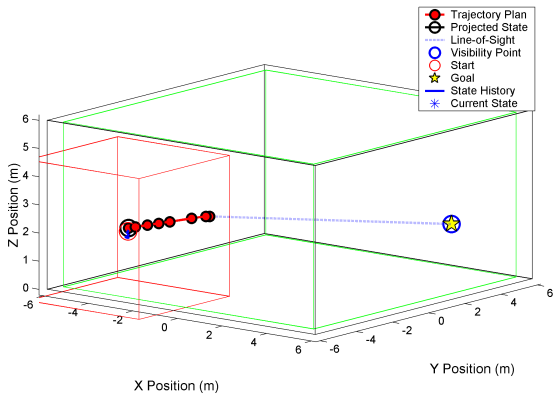
Figure 3-13: Efficient RSBK with both Selective CT and VDCS.

while the trajectory in Fig. 3-13(b) avoids it to the left. However, because the problem geometry is symmetric with respect to the axis connecting the start and goal, either direction is equally cost-effective. On the other hand, there are significant improvements in complexity obtained by using VDCS. Table 3.1 indicates that for both scenarios with VDCS added, the mean number of constraints has decreased by over 1000, while the average solution time has decreased by 12-14%.

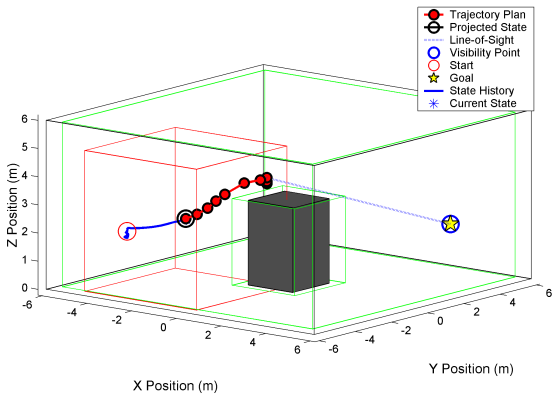
Finally, Figs. 3-14 and 3-15 show the same scenarios as Figs. 3-13(a) and 3-13(b), respectively, but with the assumption of perfect environmental knowledge removed. Instead, the vehicle must use a detection radius of 2.5 meters, indicated by a red box in the figures, to navigate and identify obstacles. As noted in Section 3.5.3, the vehicle must remain within its detection box during each Efficient RSBK optimization to ensure long-term feasibility. This is indeed the case at every timestep for these scenarios.

At the first timestep, $t = 0$ (Figs. 3-14(a) and 3-15(a)), the vehicle is not aware of the obstacle, and chooses a trajectory which unknowingly aims directly for it. If the trajectory were not constrained to stay within the detection radius, the optimal trajectory *would* be infeasible, due to the invisible obstacle it intersects. After two timesteps, the obstacle is detected in both scenarios (Figs. 3-14(b) and 3-15(b)), forcing the planner to recompute the cost-to-go map (line 6 of Algorithm 3.4) and identify a new trajectory which avoids the obstacle. Because of the changing levels of situational awareness, the resulting trajectories (Figs. 3-14(d) and 3-15(d)) are not quite as smooth as those in previous scenarios. However, all hard and soft constraints are still satisfied for both scenarios.

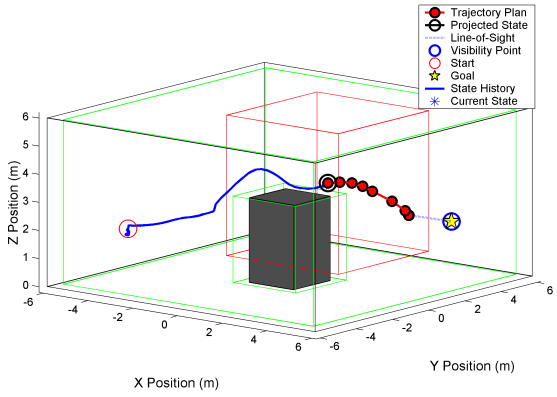
An interesting consequence of the detection radius approach is a decrease in the average solution time over the course of the trajectory, compared to scenarios involving a perfect situational awareness. In fact, both scenarios using a detection radius have solution times at least 30% faster than any other scenario considered (Table 3.1). Because the optimal trajectory *must* remain within the detection radius, the freedom of the algorithm in selecting trajectories is severely reduced. Additionally, only those obstacles in close proximity to the vehicle need to be considered, making the detection



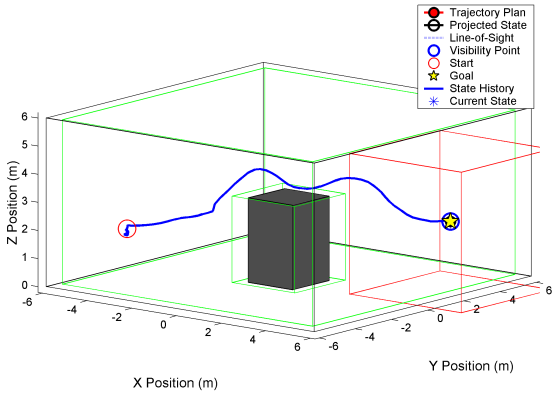
(a) Optimization 1



(b) Optimization 3

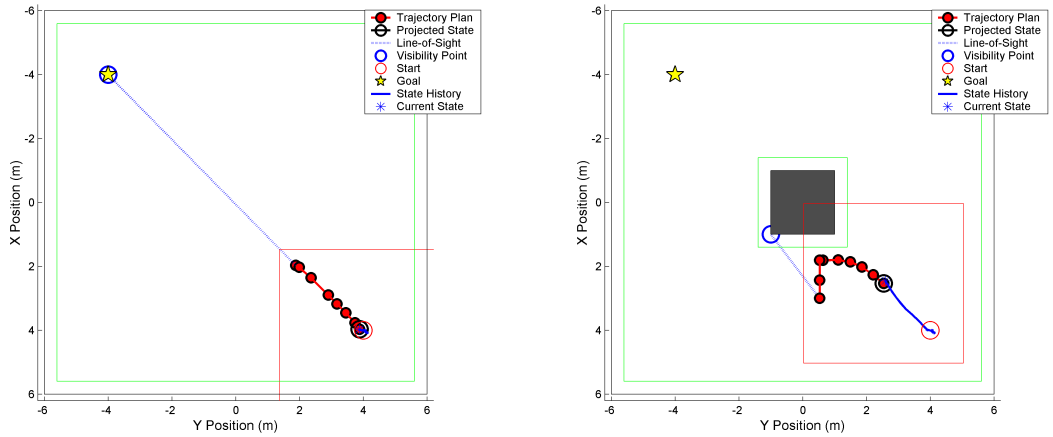


(c) Optimization 8



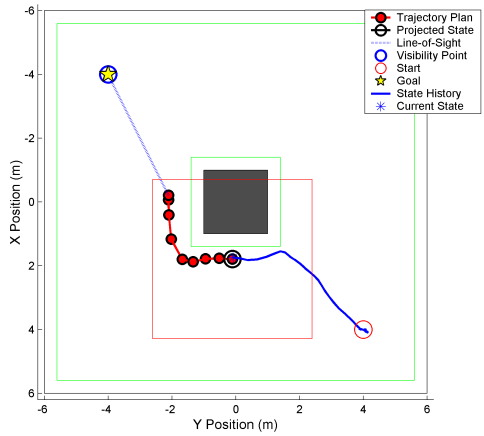
(d) Optimization 14

Figure 3-14: Flying over an obstacle using Efficient RSBK with a detection radius (red box).

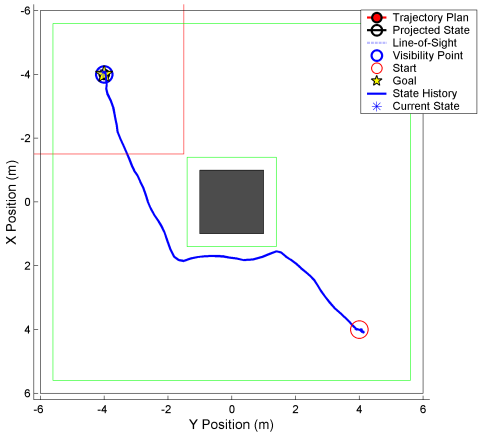


(a) Optimization 1

(b) Optimization 3



(c) Optimization 5



(d) Optimization 14

Figure 3-15: Flying around an obstacle using Efficient RSBK with a detection radius (red box).

box a *de facto* obstacle reachable horizon (Section 3.6.3). This approach can be quite useful for limiting the scope of the MILP optimization in cluttered environments, since the cost map is available to help plan long-term trajectories.

3.8 Conclusions

This chapter has presented the Efficient RSBK trajectory planner, which integrates existing and novel MILP-based planning techniques to quickly compute robust trajectories within complex and uncertain environments. Selective CT allows the set of input decision variables to cover larger planning horizons with a minimal increase in complexity, though the guarantee of an always-feasible optimization is lost. The VDCS scheme re-allocates constraints such that they are sufficient for good trajectory behavior where needed, and sparse for low complexity elsewhere. Simulation results show the advantages of incorporating both theoretically-inclined and implementation-inclined algorithms. By reducing the complexity of the MILP optimization, this planner demonstrates robustness and validates the techniques being used for efficient real-time planning.

Chapter 4

Experimental Results

In this chapter, hardware experiments are performed with quadrotors in the Real-time indoor Autonomous Vehicle test ENvironment (RAVEN) [67] to demonstrate the trajectory planner components introduced in this thesis. The flight results include tests of a novel low-level control scheme, introduced in this chapter, as well as validation of the MILP-based trajectory planner from Chap. 3, Efficient RSBK. The low-level controller includes several significant improvements to the previous design within the RAVEN software architecture which improve overall trajectory following and enable tracking of aggressive maneuvers. While the low-level controller has been designed specifically for a quadrotor vehicle, many of its design principles can be applied to other vehicle types. The objective of these flight results is to verify the goals stated in Chap. 1 are satisfied by the integrated vehicle planning and control architecture.

The structure of this chapter is as follows. Section 4.1 introduces the quadrotor hardware and establishes the system model used by the low-level controller. The RAVEN testbed is introduced in Section 4.2, while the low-level controller is presented in Section 4.3. Hardware flight demonstrations are detailed in Sections 4.4 and 4.5 for the low-level controller and Efficient RSBK, respectively. Finally, Section 4.6 offers concluding remarks.

Figure 4-1: Draganflyer Quadrotor

Figure 4-2: X-UFO Quadrotor

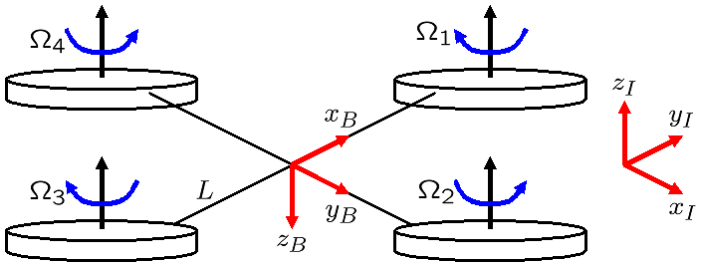


Figure 4-3: Quadrotor Model, at Zero-Angle Orientation

4.1 The Quadrotor

Though the work in this thesis is applicable to any UAV, the results in this chapter focus on the quadrotor vehicle. A quadrotor uses four rotor blades in a cross configuration to achieve hover and maneuvering with minimal gyroscopic effects. Quadrotors are desirable for smaller-scale missions due to their simple design, maneuverability, yaw-stable configuration, and vertical takeoff and landing (VTOL) capabilities [68]. The hardware results in this chapter focus on two specific quadrotors, the Draganflyer (Fig. 4-1) and the X-UFO (Fig. 4-2).

A linearized version of the quadrotor dynamics is now derived, based heavily on work in Refs. [1, 68]. Consider the model of the quadrotor in Fig. 4-3, with (x_I, y_I, z_I) defining the inertial coordinate frame and (x_B, y_B, z_B) defining the quadrotor's body axes. Here L is the maximum distance between the quadrotor's center of mass and a rotor axis, while Ω_i is the angular velocity of rotor i. Assume that the vehicle is symmetric, such that $I_{xy} = I_{xz} = I_{yz} = 0$. Finally, assume that the configuration shown in Fig. 4-3 corresponds to a roll ϕ, pitch θ, and yaw ψ all equal to zero.

Suppose the Euler angles are propagated using the small-angle assumption [1];

then the quadrotor dynamics are given by

$$\ddot{\phi} = \dot{\theta}\dot{\psi}\frac{I_y - I_z}{I_x} - \frac{J_r}{I_x}\dot{\theta}\Omega + \frac{L}{I_x}\delta_{\text{roll}}, \tag{4.1}$$

$$\ddot{\theta} = \dot{\phi}\dot{\psi}\frac{I_z - I_x}{I_y} + \frac{J_r}{I_y}\dot{\phi}\Omega + \frac{L}{I_y}\delta_{\text{pitch}}, \tag{4.2}$$

$$\ddot{\psi} = \dot{\phi}\dot{\theta}\frac{I_x - I_y}{I_z} + \frac{1}{I_z}\delta_{\text{yaw}}, \tag{4.3}$$

$$\ddot{x}_E = (\sin\phi\cos\psi - \cos\phi\sin\theta\sin\psi)\frac{1}{m}\delta_{\text{coll}}, \tag{4.4}$$

$$\ddot{y}_E = (-\sin\phi\sin\psi - \cos\phi\sin\theta\cos\psi)\frac{1}{m}\delta_{\text{coll}}, \tag{4.5}$$

$$\ddot{z}_E = -g + (\cos\phi\cos\theta)\frac{1}{m}\delta_{\text{coll}}, \tag{4.6}$$

where I_x, I_y, and I_z are the body moments of inertia, J_r is the rotor blade moment of inertia, and Ω is the net rotor angular rate,

$$\Omega = \Omega_1 + \Omega_3 - \Omega_2 - \Omega_4. \tag{4.7}$$

The inputs δ_{roll}, δ_{pitch}, δ_{yaw}, and δ_{coll} are referred to as the roll, pitch, yaw, and collective, respectively, and are related to the rotor angular rates through the non-linear relation

$$\begin{bmatrix} \delta_{\text{roll}} \\ \delta_{\text{pitch}} \\ \delta_{\text{yaw}} \\ \delta_{\text{coll}} \end{bmatrix} = \begin{bmatrix} 0 & -b & 0 & b \\ b & 0 & -b & 0 \\ d & -d & d & -d \\ -b & -b & -b & -b \end{bmatrix} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix}, \tag{4.8}$$

where b is the rotor thrust factor and d is the rotor drag factor.

To identify a simplified form of the vehicle dynamics, assume that $\Omega = 0$ and $I_x = I_y$ due to vehicle symmetry. Applying the small-angle assumption to the Euler

angles and angular rates, the vehicle dynamics (4.1)-(4.6) become

$$\ddot{\phi} = \frac{L}{I_x}\delta_{\text{roll}}, \qquad \ddot{\theta} = \frac{L}{I_y}\delta_{\text{pitch}}, \qquad \ddot{\psi} = \frac{1}{I_z}\delta_{\text{yaw}}, \tag{4.9}$$

$$\ddot{x}_E = \phi\frac{1}{m}\delta_{\text{coll}}, \qquad \ddot{y}_E = -\theta\frac{1}{m}\delta_{\text{coll}}, \qquad \ddot{z}_E = -g + \frac{1}{m}\delta_{\text{coll}}. \tag{4.10}$$

By linearizing the collective about the equivalent gravity force [68],

$$\delta_{\text{coll}} = mg + \hat{\delta}_{\text{coll}}, \tag{4.11}$$

and dropping small terms, (4.9)-(4.10) are simplified to the set of equations

$$\ddot{\phi} = \frac{L}{I_x}\delta_{\text{roll}}, \qquad \ddot{x}_E = g\phi, \tag{4.12}$$

$$\ddot{\theta} = \frac{L}{I_y}\delta_{\text{pitch}}, \qquad \ddot{y}_E = -g\theta, \tag{4.13}$$

$$\ddot{z}_E = \frac{1}{m}\hat{\delta}_{\text{coll}}, \tag{4.14}$$

$$\ddot{\psi} = \frac{1}{I_z}\delta_{\text{yaw}}. \tag{4.15}$$

Note that the dynamics have decoupled into four separate subproblems, one per row, each with its own unique input. Each component of the translational and rotational dynamics can be modeled separately as a simple double integrator. This is the model used by the low-level controller (Section 4.3) to control the UAV.

4.2 RAVEN Testbed

The Real-time indoor Autonomous Vehicle test ENvironment (Fig. 4-4) is a testbed designed for the execution of missions by teams of UAVs [1, 67, 69, 70]. By decomposing UAV missions into the hierarchy specified in Fig. 1-5, the RAVEN software architecture maintains a modular format amenable to incremental upgrades and enhancements. The software autonomously manages low-level tasks such as navigation and tasking, allowing operators to focus on higher-level objectives. The testbed is

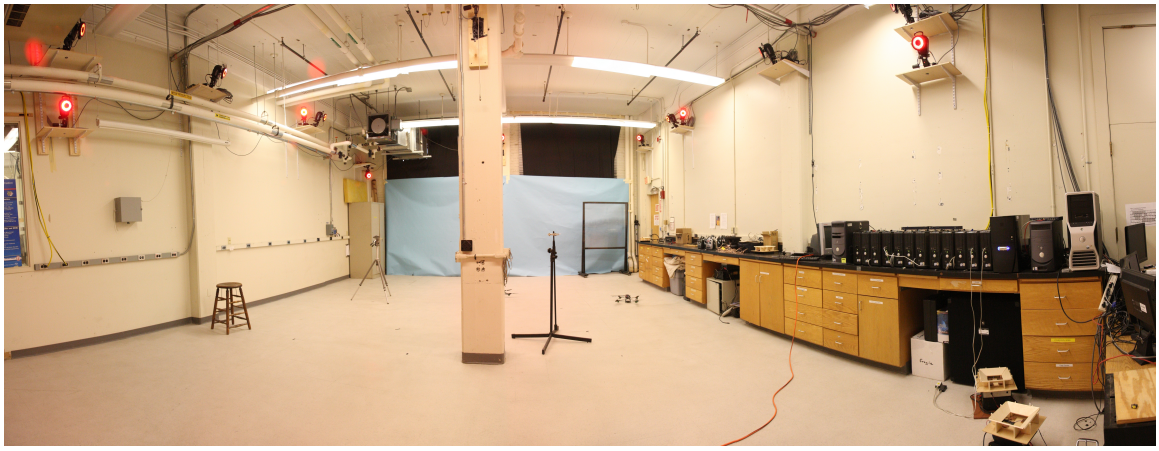


Figure 4-4: RAVEN Testbed

also outfitted with health management tools which reduce the risk of vehicle failure within the testbed [71, 72].

State data is provided to the vehicles within RAVEN by the Vicon Positioning System [73], a global metrology system consisting of infrared LED cameras and lightweight reflective markers. These markers are individually attached to vehicles (see Figs. 4-1 – 4-2) in order to create a uniquely detectable configuration for each. Eighteen cameras placed around the room (see Fig. 4-4) record the locations where light is reflected. A central processing unit then assimilates each camera's observations to identify the state of each marker configuration in the room. This state data is produced and filtered at 100 Hz and is generally quite accurate, with a static error rarely exceeding 0.4 millimeters [1]. With its non-intrusive state estimation system and controlled environment, the RAVEN testbed promotes rapid prototyping of vehicles and control algorithms.

Within the RAVEN software architecture, the Vehicle Controller (see Fig. 1-5) is further decomposed into several smaller components (Fig. 4-5). With the exception of the Reference Controller output, all inputs and outputs of the Vehicle Controller components are waypoints of some form. All waypoints include a position as well as a heading command (typically zero), but may also include a velocity command and/or arrival time. The User Interface and Vehicle Manager are discussed below; the Trajectory Generator and Reference Controller are grouped together as the *low-*

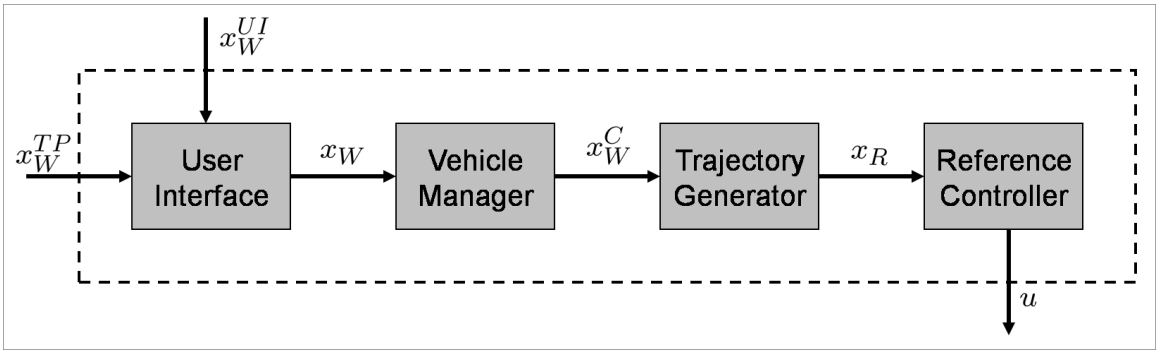


Figure 4-5: RAVEN Vehicle Controller Architecture

level controller and are discussed in detail in Section 4.3.

4.2.1 User Interface

A graphical user interface is used by the operator to monitor progress and declare tasks, either through an external planner or by the user directly. If the vehicle waypoints are provided by an external trajectory planner, such as Efficient RSBK, a "black box" within the interface forwards waypoints x_W^{TP} to the Vehicle Manager and provides Vicon state data to the external planner at 2 Hz. The user can also declare tasks directly, through the user interface's real-time graphical representation of the environment (Fig. 4-6). Within this environment, the user can select vehicles and issue vehicle-specific commands, such as waypoints x_W^{UI}.

4.2.2 Vehicle Manager

The Vehicle Manager handles many of the vehicle's administrative needs, including health and task management. The manager receives and processes task messages from the user interface, monitors the status of current tasks, and maintains a list of future tasks to be performed. Each task is associated with one or more current waypoints x_W^C which the vehicle is to approach in some sequence. Whenever a task is in progress, the manager uses a set of threshold criteria to determine when to move to the next waypoint in the queue. These criteria depend on the type of waypoint task being considered, and are discussed further in the following section.

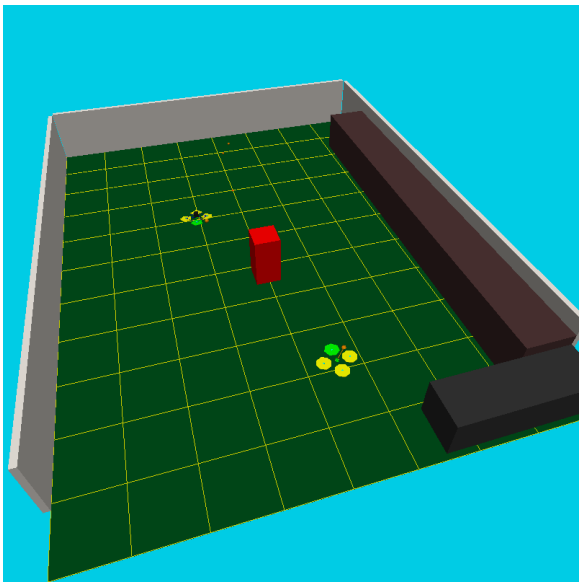


Figure 4-6: RAVEN Visualization Environment

4.3 Low-Level Controller

This section introduces a low-level control scheme designed to encourage accurate waypoint tracking and enable following of aggressive trajectories. After establishing the task termination criteria used by the Vehicle Manager (Section 4.2.2), the Trajectory Generator and Reference Controller are each discussed.

The UAV software architecture uses two specific types of waypoint tasks: *speed-based* waypoints and *time-based* waypoints. Speed-based waypoints, built into the original RAVEN architecture [1], are used by the vehicle while hovering or moving to a waypoint at a fixed speed. A speed-based waypoint is specified by a desired position $\mathbf{p}_W$, heading ψ_W, and traverse speed v_W, and is used for all tasks except fly-to-waypoint tasks. Time-based waypoints, on the other hand, dictate that the vehicle arrive at a waypoint position $\mathbf{p}_W$ with a desired velocity $\mathbf{v}_W$ at a specific arrival time Δ seconds in the future. The low-level controller for this waypoint type is a novel contribution to the software architecture, and is used whenever the vehicle is completing a fly-to-waypoint task. Note that all time-based waypoint tasks have an assumed heading $\psi_W = 0$. Finally, either type of waypoint task may include an additional command to clear the current UAV task list before being added.

4.3.1 Termination Criteria

The Vehicle Manager (Section 4.2.2) uses termination criteria, dependent on the waypoint type, to indicate when the UAV has "arrived" at each assigned waypoint. Speed-based waypoint tasks are monitored using a three-dimensional distance threshold, such that if the vehicle comes within some minimum distance of the waypoint the task is considered completed. This criterion is appropriate for speed-based waypoints because the tasks which use it typically favor arrival positional accuracy over precise following of a reference trajectory.

Since time-based waypoints are each associated with an arrival time, a task is considered complete once its arrival time has been reached, regardless of the UAV's state. With this criterion, tasks are completed at user-specified times, and rely on the low-level controller to track waypoints accurately. This condition is particularly useful when the UAV is to follow a specific sequence of waypoints. If a distance threshold criterion were to be used, a late arrival at one waypoint may invalidate the arrival times at subsequent waypoints, leading to jittery motion and possibly dangerous behavior. By making the tasks independent of the vehicle location, a sequence of waypoints can be provided to the Vehicle Controller to "pre-load" arbitrary trajectories.

4.3.2 Trajectory Generator

The Trajectory Generator uses a single input waypoint x_W^C to generate a detailed reference trajectory $x_R(t)$ for the reference controller. There are several reasons why this additional level of hierarchy is necessary for the planner. First, because the waypoint may be associated with arrival commands, such as a velocity and/or time, the waypoint alone does not provide sufficient degrees of freedom for the controller to meet the waypoint specifications. Second, this hierarchy allows the trajectory-following problem to be decoupled into two disparate components: a relatively slow ($\sim$ 1 Hz) trajectory planner which optimizes coarse trajectories in real-time, and a relatively fast (50 Hz) low-level controller which interpolates those results into a

finely-grained trajectory to track.

The trajectory generator for speed-based tasks uses a so-called "carrot controller," in which the reference position $\mathbf{p}_R$ is moved toward the goal at the specified traverse speed v_W. At each controller iteration, with a timestep spacing of δ, $\mathbf{p}_R$ is moved a distance of $v_W\delta$ in the direction $\mathbf{p}_W - \mathbf{p}$, where $\mathbf{p}$ is the UAV's current position. If the dot product $(\mathbf{p}_W - \mathbf{p}) \cdot (\mathbf{p}_W - \mathbf{p}_R)$ is negative, the waypoint has been passed and the reference is set directly to the waypoint, $\mathbf{p}_R = \mathbf{p}_W$. This approach moves the reference linearly towards the goal, and cannot explicitly incorporate an arrival velocity or time.

The trajectory generator for time-based waypoints builds a time-parametrized cubic spline connecting the initial position and velocity with the final position and velocity at the specified time. While more complex than the speed-based trajectory generator, this approach can be used to construct realistic interpretations of complex waypoint behavior, especially if the UAV needs to smoothly transition through a sequence of waypoints.

Because the spline formulation is independent and identical for all three coordinate dimensions, only the x-coordinate is described here; the y-coordinate and z-coordinate formulations follow immediately. Consider a vehicle at position p_{xI} with velocity v_{xI} which is scheduled to arrive at the position p_{xG} with velocity v_{xG} after the time interval Δ. Model the reference trajectory as a cubic function,

$$p_{xR}(t) = a_3 t^3 + a_2 t^2 + a_1 t + a_0, \tag{4.16}$$

where $a_i \ \ \forall\, i \in \mathbb{N}_3$ are coefficients to be identified. Without loss of generality, assume that the arrival time is $t = 0$; then the constraints on the reference trajectory are

$$\begin{aligned} -a_3\Delta^3 + a_2\Delta^2 - a_1\Delta + a_0 &= p_{xI}, & (4.17)\\ 3a_3\Delta^2 - 2a_2\Delta + a_1 &= v_{xI}, & (4.18)\\ a_0 &= p_{xG}, & (4.19)\\ a_1 &= v_{xG}. & (4.20) \end{aligned}$$

This system of equations can be uniquely solved for any $\Delta > 0$, yielding the coefficients

$$\begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} 2/\Delta^3 & 1/\Delta^2 & -2/\Delta^3 & 1/\Delta^2 \\ 3/\Delta^2 & 1/\Delta & -3/\Delta^2 & 2/\Delta \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} p_{xI} \\ v_{xI} \\ p_{xG} \\ v_{xG} \end{bmatrix}. \tag{4.21}$$

For this approach, only one spline needs to be considered at a time. Whenever a new task is initiated, the vehicle manager uses (4.21) to generate the spline coefficients for each coordinate dimension. For the first spline in a sequence, the initial position $\mathbf{p}_I$, velocity $\mathbf{v}_I$, and time t_I are taken from the most recent available vehicle state data. For subsequent waypoints, these quantities are assigned the position, velocity, and time of the *previous waypoint* in the list. In this manner, the time-based trajectory generator can build complex, continuously differentiable trajectories consisting of piecewise cubic interpolations. In both cases, the final position $\mathbf{p}_G$, velocity $\mathbf{v}_G$, and time t_G are taken from the waypoint task itself. Once a spline has been initialized, the trajectory generator can identify its reference position at any time by evaluating (4.16) for each dimension using the current time.

4.3.3 Reference Controller

The Reference Controller uses the reference trajectory $x_R(t)$ from the Trajectory Generator to calculate quadrotor control inputs at 50 Hz, based on the model (4.12)-(4.15). The four inputs, δ_{coll}, δ_{roll}, δ_{pitch}, and δ_{yaw}, are normalized to the range $[-1, +1]$, where 0 is the nominal trim and ± 1 are full deflections in either direction.

The RAVEN baseline controller, used for speed-based waypoints, applies LQR control to (4.12)-(4.15) with an integrator state added to each decoupled subsystem. In the LQR calculations, a relatively large cost is placed on the angular position, while a relatively small cost is placed on the angular rate [1]. The resulting control

laws are

$$\hat{\delta}_{\text{coll}} = K_{zg}\left[K_{zi}\int(p_{zR}-z)+K_{zp}(p_{zR}-z)-K_{zd}\dot{z}\right], \tag{4.22}$$

$$\delta_{\text{roll}} = K_{yg}\left[K_{yi}\int(p_{yR}-y)+K_{yp}(p_{yR}-y)-K_{yd}\dot{y}+K_{rp}\phi+K_{rd}\dot{\phi}\right], \tag{4.23}$$

$$\delta_{\text{pitch}} = K_{xg}\left[K_{xi}\int(p_{xR}-x)+K_{xp}(p_{xR}-x)-K_{xd}\dot{x}+K_{tp}\theta+K_{td}\dot{\theta}\right], \tag{4.24}$$

$$\delta_{\text{yaw}} = K_{wg}\left[K_{wi}\int(\psi_R-\psi)+K_{wp}(\psi_R-\psi)-K_{wd}\dot{\psi}\right]; \tag{4.25}$$

the gains are not edited in this thesis and thus are omitted for brevity. Note that gravity is not explicitly included in (4.22), but is balanced by integrator error buildup in (4.22) during the first moments of flight.

While this controller has been demonstrated to yield good performance for pure hover, it is not sufficient for effective tracking of more aggressive maneuvers. In (4.22)-(4.24), only the position is tracked by the controller; the velocity, Euler angles, and angular rates are all regulated to zero. This can lead to poor performance when tracking trajectories require high speeds and/or large bank angles. For this reason, the time-based reference controller modifies (4.22)-(4.24) to include velocity and Euler angle tracking.

Consider the derivatives of the reference trajectory (4.16) for the x-direction (the $y-$ and $z-$ directions are similar),

$$v_{xR}(t) = \frac{d}{dt}p_{xR}(t) = 3a_3t^2+2a_2t+a_1, \tag{4.26}$$

$$a_{xR}(t) = \frac{d^2}{dt^2}p_{xR}(t) = 6a_3t+2a_2. \tag{4.27}$$

The addition of velocity tracking to (4.22)-(4.24) is straightforward:

$$-K_{zd}\dot{z} \longrightarrow K_{zd}(v_{zR}-\dot{z}), \tag{4.28}$$

$$-K_{yd}\dot{y} \longrightarrow K_{yd}(v_{yR}-\dot{y}), \tag{4.29}$$

$$-K_{xd}\dot{x} \longrightarrow K_{xd}(v_{xR}-\dot{x}). \tag{4.30}$$

This formulation allows both waypoint types to use the same controller. For example, if a speed-based waypoint is issued, (4.22)-(4.24) are recovered by simply setting $\mathbf{v}_R = \mathbf{0}$.

For angular tracking, it is necessary to relate the pitch angle θ and roll angle ϕ with the reference acceleration terms, a_{xR} and a_{yR}. Suppose that the quadrotor intends to achieve an acceleration of a_R in some reference direction via an input acceleration a_V at the tilt angle α (Figure 4-7). In order to maintain altitude, the vehicle must satisfy the constraints

$$a_V \cos\alpha = g, \tag{4.31}$$

$$a_V \sin\alpha = a_R, \tag{4.32}$$

where g is acceleration due to gravity. Dividing (4.32) by (4.31) yields

$$\tan\alpha = \frac{a_R}{g} \quad \Rightarrow \quad \alpha = \tan^{-1}\frac{a_R}{g}. \tag{4.33}$$

Finally, the accelerations must be converted from the inertial frame (x_I, y_I) into the body frame (x_B, y_B) (Fig. 4-3), using the relation

$$\begin{bmatrix} x_B \\ y_B \end{bmatrix} = \begin{bmatrix} \sin\psi & \cos\psi \\ \cos\psi & -\sin\psi \end{bmatrix} \begin{bmatrix} x_I \\ y_I \end{bmatrix}. \tag{4.34}$$

Combining (4.33) and (4.34) yields the angle-acceleration relationships

$$\theta = -\tan^{-1}\frac{a_x}{g}, \tag{4.35}$$

$$\phi = \tan^{-1}\frac{a_y}{g}. \tag{4.36}$$

The corresponding tracking substitutions in (4.23)-(4.24) are

$$K_{tp}\theta \longrightarrow K_{tp}\left(\theta + \tan^{-1}\frac{a_{xR}}{g}\right), \tag{4.37}$$

$$K_{rp}\phi \longrightarrow K_{rp}\left(\phi - \tan^{-1}\frac{a_{yR}}{g}\right). \tag{4.38}$$

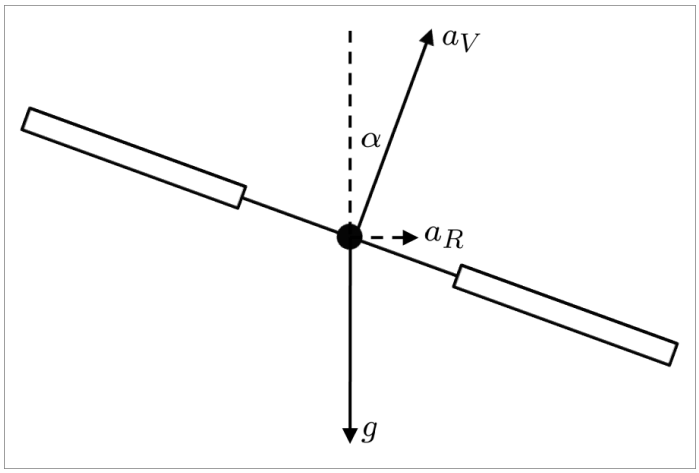


Figure 4-7: Derivation of Angle Tracking

During flight tests, it has been observed that the quadrotor often slows down faster than predicted when moving from an angle *towards* a waypoint (to speed up) to an angle *away from* a waypoint (to slow down). This loss of momentum often causes the vehicle to deviate significantly from the reference trajectory, resulting in large errors at the time of arrival (Section 4.4). To correct this, a drag feed-forward (DFF) term may be added to the reference acceleration terms,

$$a_{xR}^{ff} = K_{drag}^{x}\dot{x}^2\text{sgn}(x), \tag{4.39}$$

$$a_{yR}^{ff} = K_{drag}^{y}\dot{y}^2\text{sgn}(y), \tag{4.40}$$

where sgn$(\cdot)$ is +1 for positive inputs, -1 for negative inputs, and 0 otherwise.

Remark 4.1. (*angular rate tracking*) While angular rate tracking could also be used in (4.23)-(4.24), it is actually counterproductive for a cubic-based reference. Because the reference position (4.16) is a cubic function, the derivative of the reference acceleration (4.27) is a non-zero constant,

$$\dot{a}_{xR}(t) = 6a_3. \tag{4.41}$$

However, this reference neglects an impulse necessary at each boundary, corresponding to the discontinuities at the boundaries of (4.27). As a result, such a tracking

term actually works against the efforts of the other controller components. Given this result, and the fact that the angular rate gain is already the smallest for each control law [1], angular rate tracking is not included.

Remark 4.2. (*MILP simulation*) The simulation model used in Chapter 3 is based on the low-level design described in this section and operates at the same frequencies. The spline-based trajectory generator is essentially kept intact. The vehicle dynamics (4.12)-(4.15) are further simplified by inserting (4.35)-(4.36), applying the small-angle assumption, and removing all angular dynamics:

$$\ddot{x} = a_x, \tag{4.42}$$

$$\ddot{y} = a_y, \tag{4.43}$$

$$\ddot{z} = \frac{1}{m} a_z, \tag{4.44}$$

where the mass of the Draganflyer is used for m. The reference controller uses (4.22)-(4.24) with the angular terms removed and (4.28)-(4.30) included:

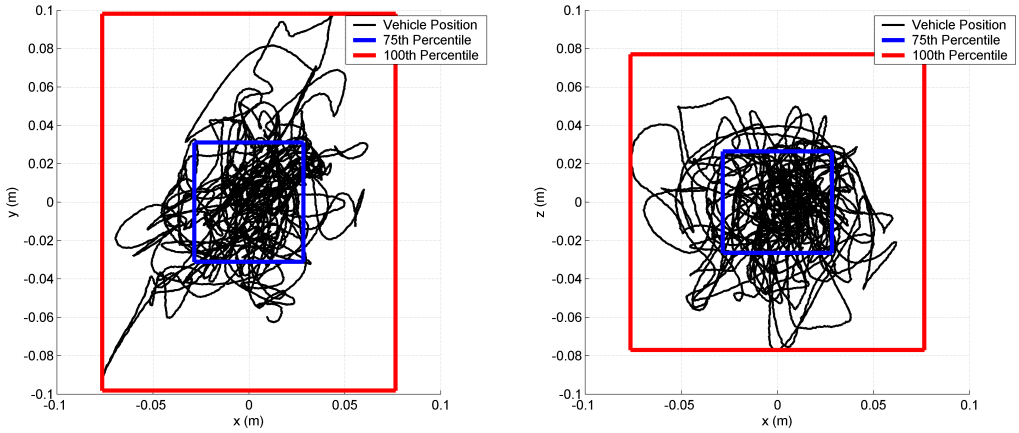
$$a_x = K_{xg}\left[K_{xi}\int (p_{xR} - x) + K_{xp}(p_{xR} - x) - K_{xd}(v_{xR} - \dot{x})\right], \tag{4.45}$$

$$a_y = K_{yg}\left[K_{yi}\int (p_{yR} - y) + K_{yp}(p_{yR} - y) - K_{yd}(v_{yR} - \dot{y})\right], \tag{4.46}$$

$$a_z = K_{zg}\left[K_{zi}\int (p_{zR} - z) + K_{zp}(p_{zR} - z) - K_{zd}(v_{zR} - \dot{z})\right]. \tag{4.47}$$

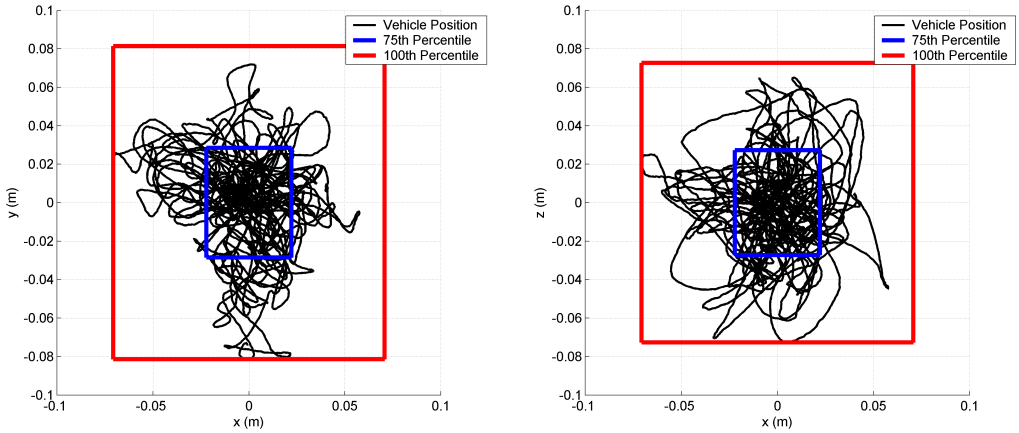
4.4 Flight Results: Low-Level Controller

In this and the following section, flight results are demonstrated using the Draganflyer and X-UFO quadrotors in the RAVEN testbed. The low-level controller of Section 4.3 is demonstrated in this section, followed by flight results using Efficient RSBK in Section 4.5.



(a) xy-plane (b) xz-plane

Figure 4-8: Draganflyer Stationary Hover Flight Test Deviations



(a) xy-plane (b) xz-plane

Figure 4-9: X-UFO Stationary Hover Flight Test Deviations

4.4.1 Stationary Hover

In the first flight test, the quadrotor is to hover at the point (0.0, 0.0, 1.5)-m for five minutes. This task uses the speed-based low-level controller, and is a useful demonstration of that controller's ability to hold a specific position. Figs. 4-8 and 4-9 show the resulting deviation in each coordinate direction for the Draganflyer and X-UFO, respectively. Error boxes are marked for the 75th and 100th percentile of deviation in each coordinate direction. Note that a similar test is performed in Ref. [1].

It is clear from Figs. 4-8 and 4-9 that the existing low-level control structure en-

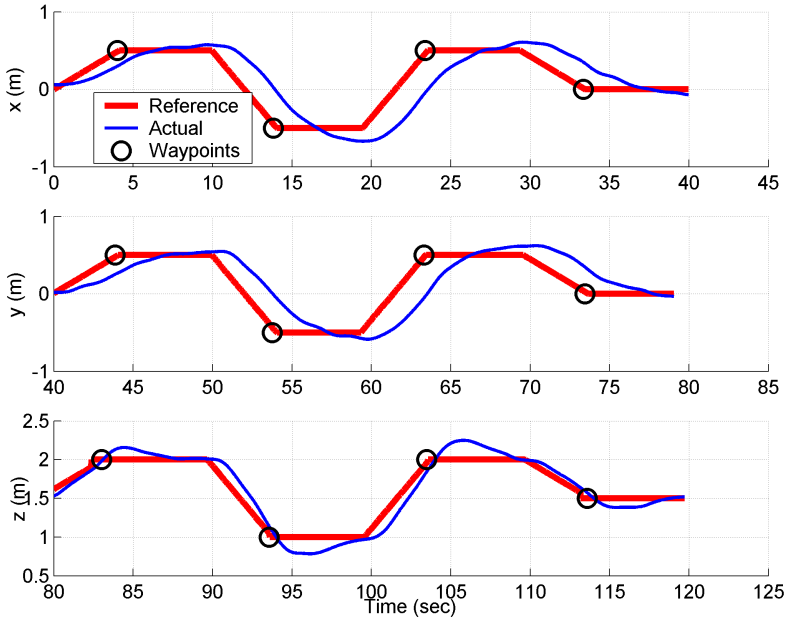
ables the quadrotors to achieve stationary hover with high accuracy. For the duration of the 5-minute flight, both vehicles remain within a 20-cm error box centered on the waypoint. Furthermore, both vehicles remain within a 7.5-cm error box for at least 75% of the flight duration. While both vehicles behave similarly, the bounding boxes for the X-UFO (Fig. 4-9) are slightly smaller than those for the Draganflyer (Fig. 4-8). This data is used in Section 4.5 to approximate the disturbance environment for these vehicles.

4.4.2 Single-Dimension Tests

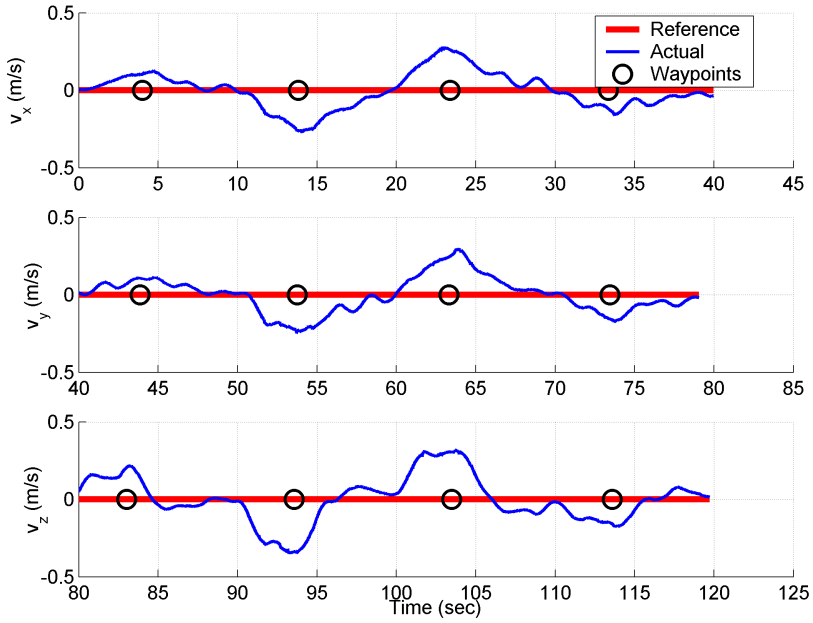
The next series of flight tests compares the effectiveness of each type of low-level control scheme following simple trajectories. In these tests, the vehicle is instructed to move back and forth along each coordinate axis, over distances not exceeding 1 meter. A new waypoint task is sent to the vehicle every ten seconds; each waypoint task is to be performed over a time interval of $\Delta = 4$ seconds, followed by a 6-second rest period. Since this arrival time is *not* compatible with the speed-based controller, its waypoints are sent to the vehicle manually, with traverse speeds v_W chosen such that the reference arrives at the waypoint approximately 4 seconds after the task is initiated.

Fig. 4-10 shows the reference and actual position and velocity profiles for the Draganflyer quadrotor using the speed-based controller. First, observe that while the position reference consists of straight-line segments, the velocity reference remains at zero throughout all trajectories. This is accurate, since the speed-based controller regulates all velocity components to zero. However, these two references are contradictory, and work against each other during maneuvers.

As a result, the vehicle tends to lag significantly behind the reference position profile (Fig. 4-10(a)) for most waypoints. In some instances, the reference arrives at the waypoint before the vehicle has covered even half the distance; the vehicle then may not arrive for up to 4 more seconds. This is supported by the vehicle's velocity profile (Fig. 4-10(b)). For the tasks where the vehicle is to maneuver 1 meter in 4 seconds, the desired traverse speed is 0.25 m; however, the velocity profile only



(a) Position Profile



(b) Velocity Profile

Figure 4-10: Position and velocity profile for the Draganflyer quadrotor following a simple trajectory using the speed-based low-level controller. Note the times - each dimension is tested individually and in succession.

reaches this speed around the time when it should already be at the goal.

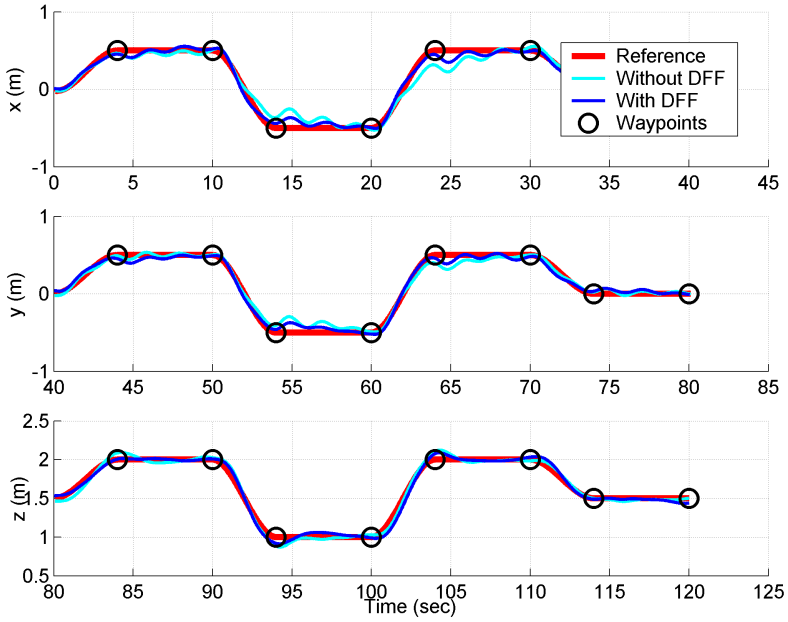
The z-controller, on the other hand, does achieve better performance compared to the other dimensions. While the vehicle still exhibits some lag and overshoots most waypoints, the vehicle arrives at each waypoint at or before the designated time. Additionally, its velocity profile remains relatively static during each task, near the desired traverse speed.

Fig. 4-11 shows the same results for the Draganflyer quadrotor, but using the *time-based* controller either with or without DFF. Here, the position reference is a continuously differentiable curve, with a cubic spline preceding each waypoint. (An additional waypoint is placed just before the beginning of each maneuver, instructing the vehicle to "hold position" for 6 seconds.) Additionally, the velocity reference is now a non-zero quadratic profile during each maneuver, rather than zero.

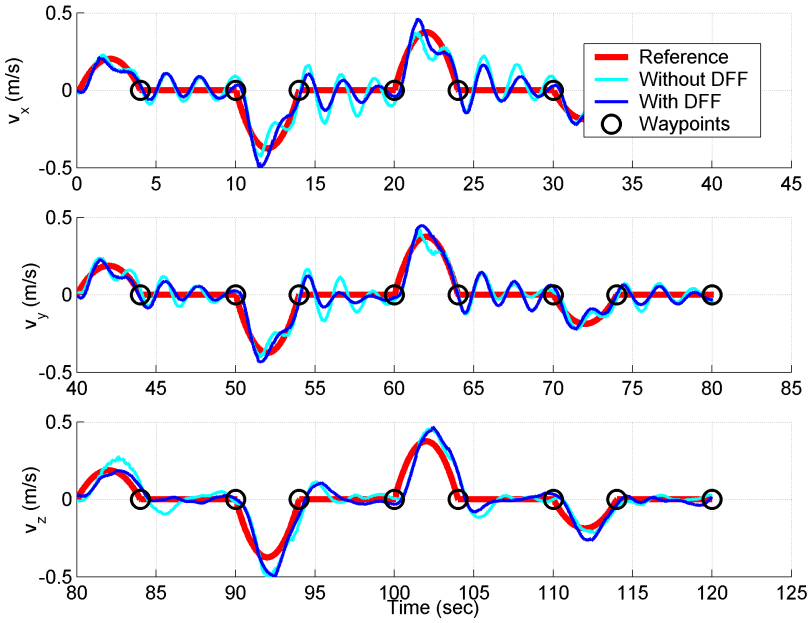
It is immediately clear in Fig. 4-11(a) that position tracking has improved significantly, compared to results with the speed-based controller. Regardless of whether DFF is used, the vehicle is within 20 cm of each waypoint at the designated arrival time. Once the arrival time has passed, very little overshoot is observed. Indeed, the only trajectory with any significant overshoot is the z-controller, though its overshoot is significantly smaller here than in Fig. 4-10(a). However, the z-controller also demonstrates the best arrival accuracy: the vehicle is within 12 cm of each waypoint at the designated arrival time.

A notable concern in the flight results for the x- and y-position without DFF is that the position tends to lag behind the reference during the second half of the maneuver, causing the vehicle to undershoot waypoints. Fig. 4-11(b) demonstrates a consistent reduction in speed which occurs whenever the vehicle begins to decelerate. This suggests that the vehicle actually slows down faster than predicted by the approximate vehicle model (4.12)-(4.15), and motivates the usage of the drag feed-forward terms (4.39)-(4.40). Fig. 4-11(a) demonstrates that by using DFF with $K_{drag} = 1.0$, the arrival accuracy is significantly improved, with the vehicle arriving within 6 cm of each waypoint at the designated arrival time.

Finally, Fig. 4-12 shows the same test as Fig. 4-11, but using the X-UFO with



(a) Position Profile



(b) Velocity Profile

Figure 4-11: Position and velocity profile for the Draganflyer quadrotor following a simple trajectory using the time-based low-level controller, both with and without drag feed-forward.

the time-based controller. By comparison to Fig. 4-11, the position tracking, and especially the velocity tracking, are superior. Using the same drag feedforward as with the Draganflyer, i.e. $K_{drag} = 1.0$, some overshoot is observed in the x- and y-position. However, for the purposes of this thesis, this result is often a preferable alternative to undershooting position waypoints. For this reason, all maneuvers performed in subsequent flight results, including Section 4.5, use the time-based controller with the drag feed-forward term.

4.4.3 Advanced Trajectories

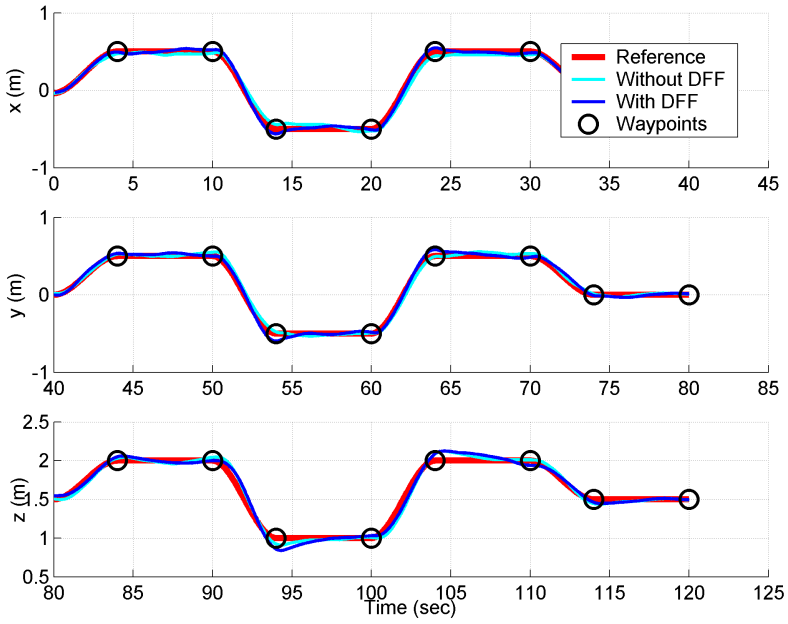
The time-based low-level controller also demonstrates strong tracking performance for waypoint sequences which require high speeds and/or complex trajectories to be successfully navigated. A key advantage of the time-based controller is the ability to build arbitrary smooth reference curves by sending the vehicle controller waypoints sampled from the curves. The effectiveness of this approach in constructing complex maneuvers is demonstrated in the following flight results.

Suppose the vehicle is to fly on a circular path with a radius of R meters and a period of T seconds. This trajectory can be approximated for the time-based low-level controller by sampling N equally-spaced waypoints around the circle, with arrival times spaced T/N seconds apart and a velocity of magnitude

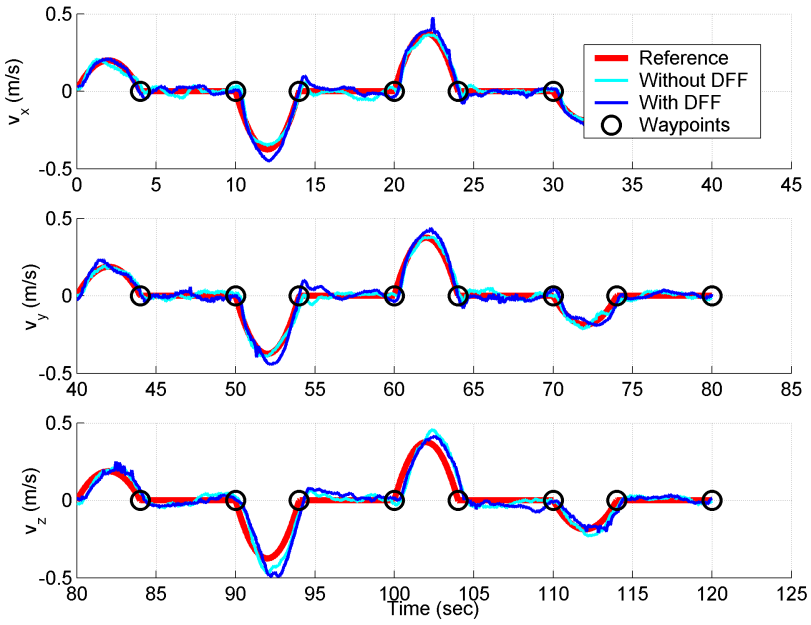
$$V_C = \frac{2\pi R}{T} \tag{4.48}$$

tangent to the circle at that point. Since cubic segments are used to interpolate between these waypoints, accurate approximations of a circular path can be achieved for small values of N. Fig. 4-13 compares the approximate reference circle ($N = 4$) used below to the exact circle it represents.

Fig. 4-14 shows the reference and actual trajectory for an X-UFO using the time-based low-level controller to fly three circular laps in RAVEN [70]. Each circle is centered at the point (0.0, 3.3, 1.5)-m, with $R = 1.5$, $N = 4$, and $T = 24$ (Fig. 4-14(a)) or $T = 15$ (Fig. 4-14(b)). Because the reference trajectory assumes the vehicle



(a) Position Profile



(b) Velocity Profile

Figure 4-12: Position and velocity profile for the X-UFO quadrotor following a simple trajectory using the time-based low-level controller, both with and without drag feedforward.

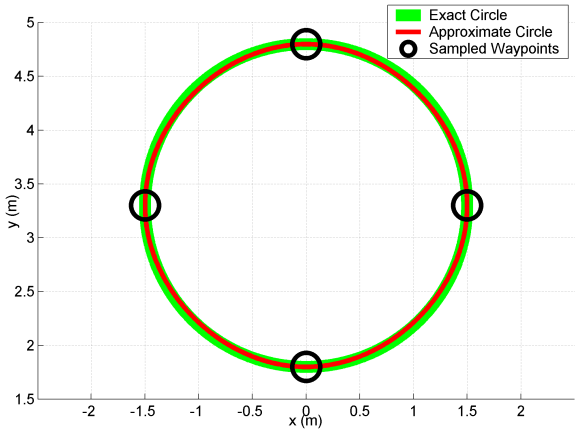
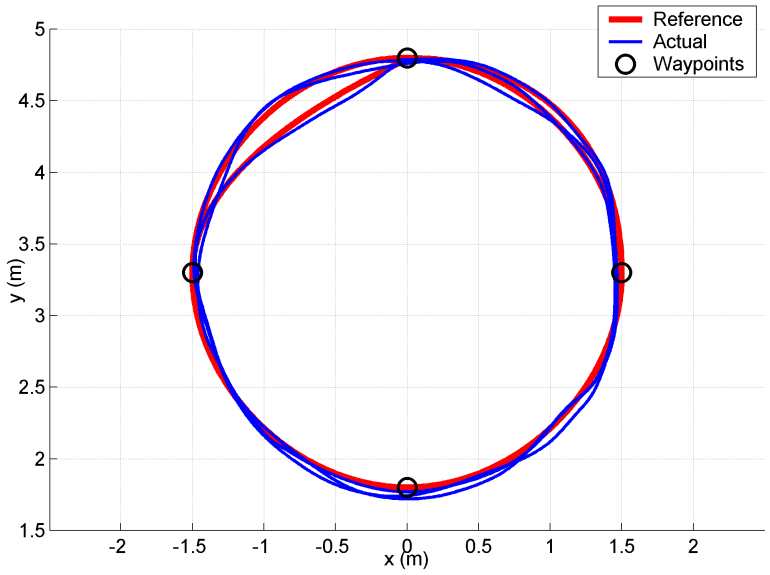


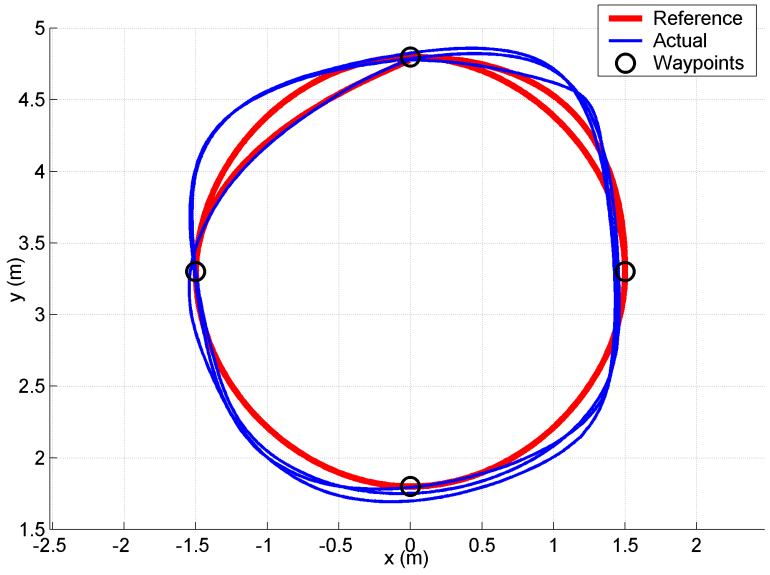
Figure 4-13: Spline-based Circle Approximation, $N = 4$

begins and ends with a zero velocity, some of the reference trajectory components at the top of each figure compensate by "pulling" the reference trajectory within the circle there. For $T = 24$ (Fig. 4-14(a)), the controller exhibits excellent tracking performance, with a maximum deviation of 10 cm. As the period decreases to $T = 15$ (Fig. 4-14(b)), tracking performance does begin to degrade. However, even in this case, the vehicle is able to approximate a circular trajectory with a relatively high speed ($V_C = 0.63$ m/s). In both cases, the vertical z-deviation never exceeds 6 cm during any of the 3 laps.

Much of this tracking performance is maintained even if the quadrotor is moving in all three dimensions. Fig. 4-15 shows the flight results for the X-UFO flying in a circle with $R = 1$, $N = 4$, and $T = 16$, but tilted 45 degrees about the $-x$-axis. The resulting tracking accuracy for this trajectory is similar to the circle in Fig. 4-14(b), which has a slightly shorter period and a larger radius. Finally, Fig. 4-16 shows the reference and actual trajectories for the Draganflyer flying a complex trajectory, involving two revolutions of a helix followed by a vertical descent down the middle. Note that in this figure, the reference trajectory is bounded from above at a height of 2.5 meters, due to the physical boundaries of the RAVEN room as specified in the trajectory generator. Nonetheless, the influence of the velocity tracking term in (4.22) is significant enough that the quadrotor's altitude exceeds 2.5 meters anyway,



(a) $T = 24$ sec



(b) $T = 15$ sec

Figure 4-14: X-UFO following a circular reference trajectory for 3 laps using the time-based low-level controller. The circle is centered at (0.0, 3.3, 1.5) m with a radius of 1.5 meters.

in order to achieve the downward velocity of the next waypoint. This, in particular, emphasizes how the additional tracking terms (4.28)-(4.30) assist in the tracking of aggressive maneuvers.

4.5 Flight Results: Efficient RSBK

This section presents flight results using the X-UFO quadrotor in RAVEN with the Efficient RSBK algorithm, integrated with the low-level controller proposed in this chapter. The intent of these results is to show that even with the advanced logic built into the Efficient RSBK algorithm, the approach remains capable of effective real-time performance in an actual implementation.

The speed-based low-level control formulation has two major limitations which make it unsuitable for the Efficient RSBK implementation. In both simulation and hardware experiments, the vehicle response time is sufficiently slow that the vehicle only ever achieves a fraction of the commanded speed. Additionally, there is no system in place to ensure the vehicle remains on schedule, making MILP replanning extremely difficult to properly execute. Indeed, the time-based controller does a superior job recovering the "intent" of the MILP planner's discrete waypoints by weaving them together into a smooth, well-defined reference trajectory.

In this scenario, the vehicle is instructed to move back and forth between two waypoints spaced 3.5 meters apart and separated by an obstacle of size $0.5 \times 2.5 \times 2.0$ m^3. The vehicle's shortest path takes it over the obstacle; however, an altitude penalty may be imposed at 2 meters, forcing the X-UFO to navigate around the obstacle. Perfect knowledge of the environment is assumed, and the VDCS scheme introduced in Fig. 3-13 is used.

Many of the parameters specified in Section 3.7 are also used here. However, the position, velocity, and acceleration constraint bounds have changed, as well as the

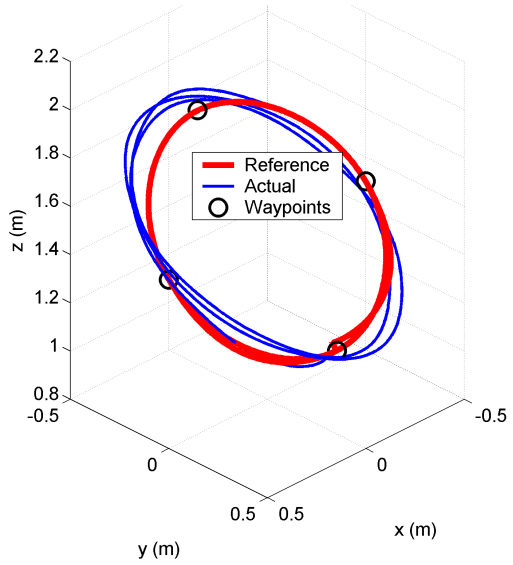


Figure 4-15: X-UFO following a circular reference trajectory tilted 45 degrees out of the xy-plane.

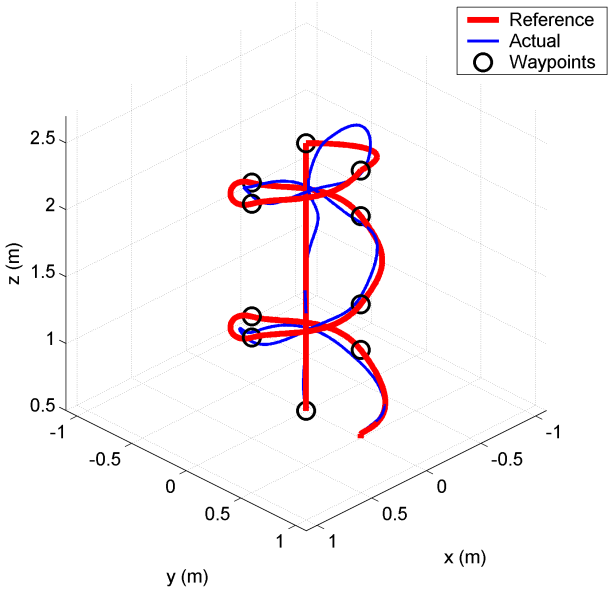


Figure 4-16: Draganflyer following a complex trajectory, including two revolutions of a helix and a vertical descent.

timestep formulation:

$$\mathbf{p}_{\min} = \begin{bmatrix} -2.5 & -2.5 & 0 \end{bmatrix}^T, \qquad \mathbf{p}_{\max} = \begin{bmatrix} 2.5 & 2.5 & 3 \end{bmatrix}^T,$$
$$\mathbf{v}_{\min} = \begin{bmatrix} -0.4 & -0.4 & -0.4 \end{bmatrix}^T, \qquad \mathbf{v}_{\max} = \begin{bmatrix} 0.4 & 0.4 & 0.4 \end{bmatrix}^T,$$
$$\mathbf{a}_{\min} = \begin{bmatrix} -0.4 & -0.4 & -0.4 \end{bmatrix}^T, \qquad \mathbf{a}_{\max} = \begin{bmatrix} 0.4 & 0.4 & 0.4 \end{bmatrix}^T,$$
$$V_{\max} = 0.4, \qquad A_{\max} = 0.4,$$
$$\delta = 2, \qquad N = 7,$$
$$N_C = \{0, 1, 2, 3, 4, 5, 6, 7\}.$$

All other parameters are the same as in Section 3.7. In fact, the disturbance level γ in that section was chosen experimentally to approximate the disturbance environment indicated by Figs. 4-8 – 4-9 for the quadrotors.

Fig. 4-17 shows the results from a single flight of the X-UFO in RAVEN, in which the Efficient RSBK algorithm is used to guide the UAV between the two waypoints. When moving from the right waypoint to the left waypoint, the aforementioned altitude penalty is enforced, while the penalty is relaxed when moving in the opposite direction. Each type of maneuver was performed five times, resulting in five "loops" moving around the obstacle.

As Fig. 4-17 indicates, the combined Efficient RSBK / low-level controller implementation is capable of following high-performance, robustly feasible trajectories. For both scenarios, the planner is able to not only identify the optimal trajectory, but also follow it with reasonable accuracy. The maneuver over the obstacle is particularly consistent, possibly due to superior tracking accuracy in the z-direction. The average optimization solve time is 364 ms over the obstacle and 502 ms around it, demonstrating a capability for on-line rapid response to a dynamic environment.

Finally, Fig. 4-18 shows the actual motion of the X-UFO within the RAVEN testbed, moving over and around the obstacle.

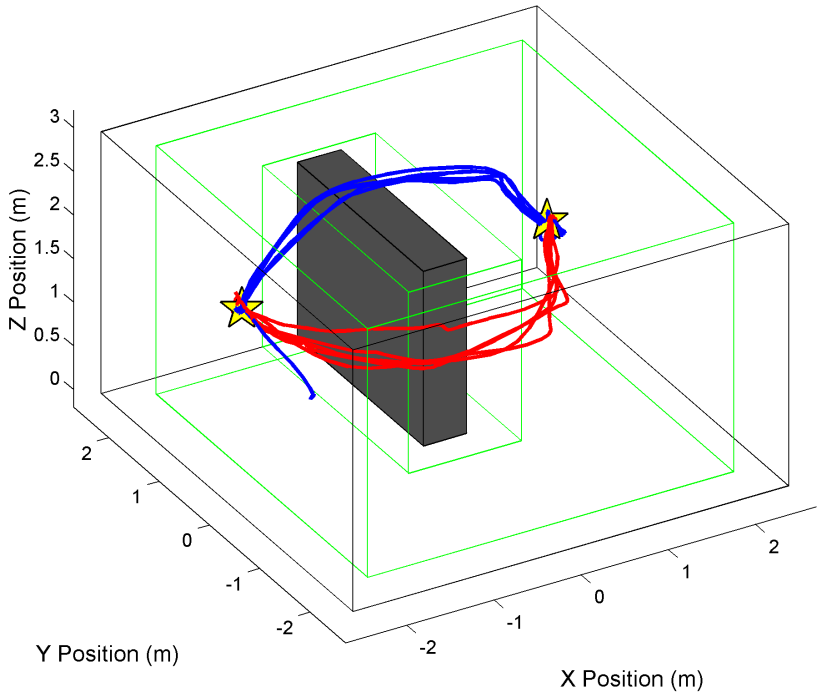


Figure 4-17: Ten applications of the Efficient RSBK algorithm on the X-UFO in a single flight. The UAV is instructed to alternate between two waypoints (yellow), flying either above (blue) or around (red) the obstacle.

4.6 Conclusions

The flight results in this chapter demonstrate that the MILP-based planning and control architecture developed throughout this thesis can be successfully implemented in a realistic hardware environment. The Efficient RSBK algorithm cannot perform its optimization sufficiently fast to directly feed waypoints into the vehicle controller; thus it is crucial that the low-level controller be able to reasonably interpolate and follow the planner's waypoint sequence. The time-based low-level controller in this chapter is designed for this purpose. Flight results show that this controller is capable of guiding a UAV through a complex sequence of waypoints with accurate position tracking, velocity tracking, and arrival times. Further flight tests integrating this low-level control scheme with Efficient RSBK verify that this planner can identify and track robustly feasible trajectories in a real-world, uncertain environment.

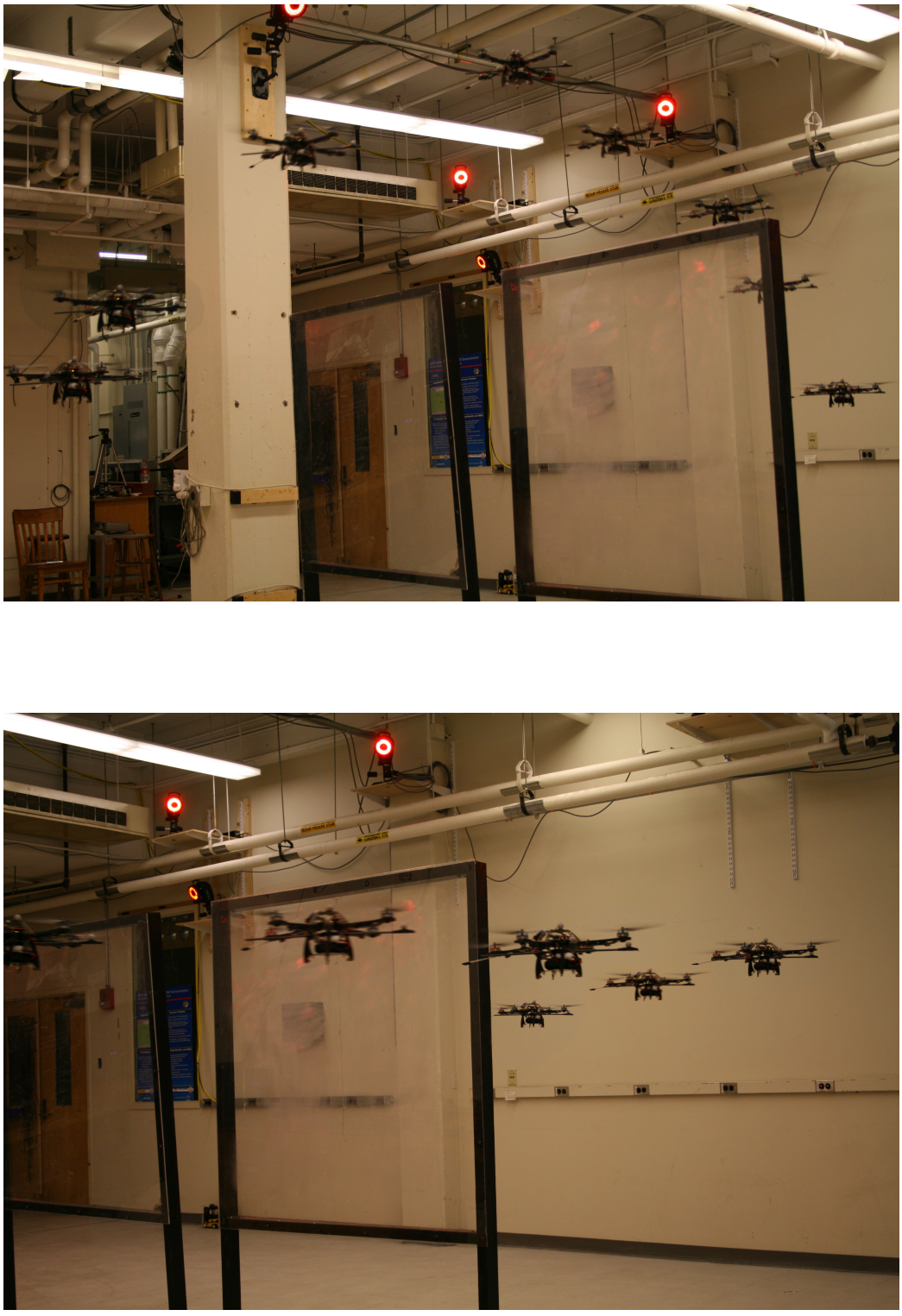


Figure 4-18: Composite photos showing sample trajectories followed by the X-UFO using Efficient RSBK.

Chapter 5

Conclusion

This thesis has presented a UAV trajectory planner which integrates existing technologies with novel refinements to identify robust, efficient trajectories in complex and uncertain environments. This planner, named Efficient RSBK, is guaranteed to satisfy all constraints at every timestep, yet minimizes the problem complexity by including only those core components necessary to maintain this robust feasibility. The effectiveness of this approach in identifying and following optimal trajectories has been thoroughly demonstrated, both in simulation and in an actual implementation using quadrotors in the RAVEN testbed.

Each chapter has explored a specific component of this planner which is necessary for trajectory planning in realistic scenarios. Chapter 2 explored a collection of techniques in RMPC, a means for guaranteeing robust feasibility in the presence of disturbances. Theoretical analysis has shown that these approaches, which are tractable for on-line optimization, are actually quite similar in form, with a key difference being the number of available on-line decision variables. Simulation results suggest that the CT formulation possesses a runtime advantage by maintaining the complexity of nominal MPC, while the AFP formulation holds an advantage in closed-loop performance *if* paired with an expected objective function.

Chapter 3 detailed the MILP-based planner, Efficient RSBK, which encodes the UAV problem constraints and identifies a coarse, high-level trajectory plan, given an input waypoint. By incorporating the RSBK algorithm, Efficient RSBK robustly sat-

isfies constraints with a guarantee of trajectory safety, while techniques from Variable MILP allow problems with long planning horizons to be modeled using a small number of degrees of freedom. Novel refinements, such as Selective CT and VDCS, give the operator freedom to place constraints and decision variables only where needed in the problem formulation.

Finally, Chapter 4 introduces a low-level planner which ensures that the intent of Efficient RSBK's output waypoint plans is actually carried out by the UAV, through reference-tracking accuracy. Its trajectory generator interpolates between sparse waypoints to create a high-fidelity reference trajectory, while the reference controller includes velocity and even angular tracking to achieve all intended maneuvers.

The viability of this approach is most easily demonstrated by verifying that the success criteria established in Section 1.3.2 are being met. Simulation and hardware results have shown that whenever the optimization can accurately predict the state's future location, all hard and soft constraints are satisfied at all desired timesteps, regardless of the disturbance realization (#1). These trajectories exhibit "expected" behavior, in the sense that they follow logical, shortest-path approaches to waypoints with minimal deviation (#2). With all refinements in place, the trajectory planning algorithm has been shown to achieve solution times at or below the chosen timestep length, indicating viability for real-time planning (#3). Once the planner has selected a direction to guide the vehicle, it maintains that direction until new obstacles are perceived (#4). Finally, Sections 4.4-4.5 have demonstrated that it is possible to accurately follow complex trajectories generated by the trajectory planner (#5).

In summary, the planner developed in this thesis demonstrates a growing level of maturity in the fields of RMPC and MILP-based trajectory planning, which will continue to enhance the autonomous capabilities of UAVs and expand the scope of their missions.

5.1 Future Work

Several key issues and topics have emerged from this work which merit further research, and are each discussed in turn below.

5.1.1 Disturbance-Aware Cost-to-go

Since a key advantage of RMPC is the conversion of an infinite-horizon control problem into a tractable finite-horizon control problem, the selection of an accurate cost-to-go is of critical importance. As described in Chap. 2, many modern RMPC approaches still use conventional forms of the cost-to-go which disregard disturbances and constraints, but can be used to satisfy some stability criteria. However, without an accurate model of the cost-to-go, the predicted cost may not accurately represent the actual incurred cost (Section 2.5.3).

A simple approximation of the cost-to-go which incorporates some notion of the disturbance environment would be a useful tool for future RMPC applications. Some preliminary analysis of this type has been performed in the Chap. 2 Appendix. There, results suggest that the conventional (e.g. quadratic) form of the cost-to-go is sufficient for the disturbance free and expected objectives, if the terminal control law is assumed to be applied at all timesteps beyond the horizon N. However, a broader analysis should be performed which considers arbitrary forms of terminal control and explores alternative ways to compute the cost-to-go, such as dynamic programming and multi-parametric quadratic programming. A truly useful approximation should incorporate some knowledge of disturbances and constraints, without significantly increasing the complexity of the *on-line* optimization.

5.1.2 Expansion of RMPC Analysis

While the theoretical and numerical analysis in Chap. 2 has offered several insights relating CT and AFP approaches, it is relatively narrow in scope: only affine feedback policies are considered. While affine approaches provide a useful balance of on-line feedback design with problem complexity, there are other available approaches

which are similarly useful. A broader analysis might incorporate several other modern RMPC approaches of comparable complexity, such as triple mode RMPC [74] and tubes [75]. Additional benchmarks may also be applied to identify performance bounds, such as minimax MPC [17]. By identifying the fundamental theoretical differences separating available approaches, it may be possible to establish a core set of RMPC principles which enables it to reach a comparable level of maturity with nominal MPC [13].

5.1.3 Intelligent Selection of Cost-to-Go Nodes

Because the shortest path between two points in a non-convex environment does not generally pass through obstacle vertices [64], the original extension of the MILP visibility graph cost-to-go [42] to three dimensions places a graph node at each obstacle edge midpoint [45]. While adding additional nodes to the visibility graph does not necessarily lead to significant improvements in accuracy [45], it may be possible to increase the utility of these nodes by reallocating their locations based on the locations of the start and goal. For example, by sampling planes which contain both the start and goal, obstacle edges which intersect these planes are good candidates for visibility graph nodes, since the shortest path might reasonably pass through or near these points. Fig. 5-1 shows a possible allocation of visibility graph nodes for an obstacle intersected by two sample planes.

5.1.4 Aerobatic Quadrotor Control

Flight experiments in RAVEN have shown that if the circle test in Fig. 4-14 is performed for $T = 8$ seconds, the motion is sufficiently quick to drive the vehicle unstable. However, the specific demands on the quadrotor at any one time instant are within the vehicle's capabilities: a speed of 1.18 m/s and a bank angle of 5.4 degrees. For more aggressive maneuvers, the simplifying small-angle assumptions used in Section 4.1 to allow LQR control may no longer be accurate. In this case, an alternative, possibly non-linear, controller may be preferable.

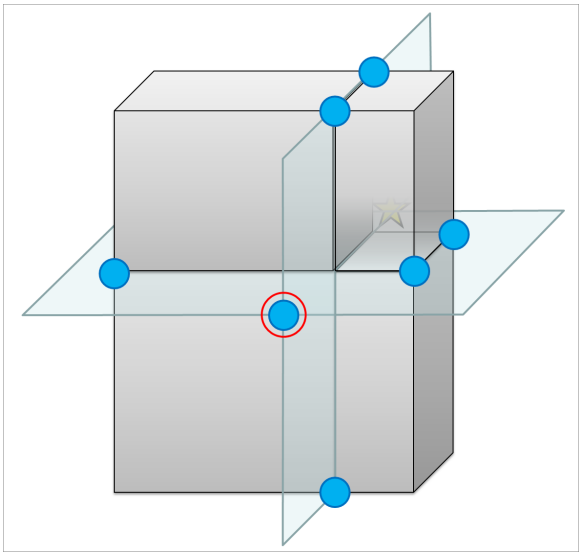


Figure 5-1: Possible allocation of obstacle nodes for the cost-to-go visibility graph, based on the locations of the start (red circle) and goal (yellow star, behind obstacle).

There are several ways in which the time-based low-level controller (Section 4.3) might be modified to enable more aerobatic maneuvers. Heading tracking may be useful in improving trajectory feasibility; in the circle test, for example, the quadrotor may achieve better performance by maintaining a fixed heading with respect to the center of the circle. Advanced controllers which could be considered include non-linear control and adaptive control, as well as hybrid control, which could augment the existing controller with a library of pre-programmed, agile maneuvers [35, 36].

Bibliography

[1] M. Valenti. *Approximate Dynamic Programming with Applications in Multi-Agent Systems*. PhD thesis, Massachusetts Institute of Technology, May 2007. 11, 19, 20, 112, 114, 115, 117, 120, 124, 125

[2] Northrop Grumman Integrated Systems. RQ-4 Block 10 Global Hawk. Online `http://www.is.northropgrumman.com/systems/ghrq4a.html`, April 2008. 17

[3] General Atomics Aeronautical Systems. Mariner. Online `http://www.ga-asi.com/products/mariner.php`, April 2008. 17

[4] Composite Engineering Inc. AQM-37: Supersonic target. Online `http://www.compositeeng.com/uav_AQM37.htm`, April 2008. 17

[5] J. S. McCarley and C. D. Wickens. Human factors concerns in UAV flight. Technical report, Institute of Aviation, Aviation Human Factors Division, University of Illinois at Urbana-Champaign, 2004. Produced for the Federal Aviation Administration. 19

[6] A. Richards, J. Bellingham, M. Tillerson, and J. P. How. Coordination and control of multiple UAVs. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Monterey, CA, August 2002. AIAA Paper 2002-4588. 21, 24

[7] A. Richards and J. P. How. A decentralized algorithm for robust constrained model predictive control. In *Proceedings of the IEEE American Control Conference*, Boston, MA, June-July 2004. 21

[8] A. Richards and J. P. How. Decentralized model predictive control of cooperating UAVs. In *Proceedings of the IEEE Conference on Decision and Control*, pages 4286–4291, Paradise Island, Bahamas, December 2004. 21, 25, 73

[9] T. Schouwenaars, J. P. How, and E. Feron. Decentralized cooperative trajectory planning of multiple aircraft with hard safety guarantees. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Providence, RI, August 2004. AIAA Paper 2004-5141. 21, 25, 73

[10] Y. Kuwata, A. Richards, T. Schouwenaars, and J. P. How. Decentralized robust receding horizon control for multi-vehicle guidance. In *Proceedings of the IEEE American Control Conference*, pages 2047–2052, June 2006. 21, 25, 28, 67, 73

[11] J. T. Betts. Survey of numerical methods for trajectory optimization. *Journal of Guidance, Control, and Dynamics*, 21(2):193–207, March-April 1998. 21

[12] C. E. García, D. M. Prett, and M. Morari. Model predictive control: Theory and practice - a survey. *Automatica*, 25(3):335–348, May 1989. 21

[13] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36:789–814, 2000. 21, 22, 35, 40, 48, 142

[14] H. Genceli and M. Nikolaou. Robust stability analysis of constrained l_1-norm model predictive control. *AIChE Journal*, 39(12):1954–1965, December 1993. 22

[15] A. Zheng and M. Morari. Robust control of linear time-varying systems with constraints. In *Proceedings of the IEEE American Control Conference*, pages 2416–2420, Baltimore, MD, June 1994. 22, 23

[16] A. Bemporad and M. Morari. Robust model predictive control: A survey. In *Robustness in Identification and Control*, volume 254, pages 207–226. Lecture Notes in Control and Information Sciences, 1999. 22

[17] P. O. M. Scokaert and D. Q. Mayne. Min-max model predictive control for constrained linear systems. *IEEE Transactions on Automatic Control*, 43(8):1136–1142, August 1998. 22, 45, 142

[18] J. R. Gossner, B. Kouvaritakis, and J. A. Rossiter. Stable generalized predictive control with constraints and bounded disturbances. *Automatica*, 33(4):551–568, 1997. 23, 35

[19] L. Chisci, J. A. Rossiter, and G. Zappa. Systems with persistent disturbances: Predictive control with restricted constraints. *Automatica*, 37:1019–1028, 2001. 23, 35, 37

[20] A. G. Richards. *Robust Constraint Model Predictive Control*. PhD thesis, Massachusetts Institute of Technology, February 2005. 23, 47, 49

[21] A. G. Richards and J. P. How. Robust stable model predictive control with constraint tightening. In *Proceedings of the IEEE American Control Conference*, pages 1557–1562, Minneapolis, MN, June 2006. 23, 25, 35, 36, 37

[22] Y. Kuwata, A. Richards, and J. How. Robust receding horizon control using generalized constraint tightening. In *Proceedings of the IEEE American Control Conference*, pages 4482–4487, New York City, NY, July 2007. 23, 25, 29, 31, 35, 36, 37, 43, 46, 78

[23] Y. Kuwata. *Trajectory Planning for Unmanned Vehicles using Robust Receding Horizon Control*. PhD thesis, Massachusetts Institute of Technology, February 2007. 23, 25, 37, 67, 72, 73, 75, 80

[24] M. V. Kothare, V. Balakrishnan, and M. Morari. Robust constrained model predictive control using linear matrix inequalities. *Automatica*, 32(10):1361–1379, 1996. 23, 45

[25] Y. I. Lee and B. Kouvaritakis. Constrained receding horizon predictive control for systems with disturbances. *International Journal of Control*, 72(11):1027–1032, 1999. 23

[26] J. Löfberg. *Minimax approaches to robust model predictive control*. PhD thesis, Linköping University, 2003. 23, 30, 35, 38, 40, 45

[27] A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski. Adjustable robust solutions of uncertain linear programs. *Mathematical Programming*, 99(2):351–376, 2004. 23, 38

[28] P. J. Goulart, E. C. Kerrigan, and J. M. Maciejowski. Optimization over state feedback policies for robust control with constraints. *Automatica*, 42:523–533, 2006. 23, 29, 31, 35, 36, 38, 39, 40, 43, 60

[29] P. J. Goulart and E. C. Kerrigan. Robust receding horizon control with an expected value cost. Submitted to Automatica, 2007. 23, 30, 35, 44

[30] T. Schouwenaars, B. de Moor, E. Feron, and J. P. How. Mixed integer programming for multi-vehicle path planning. In *Proceedings of the European Control Conference*, pages 2603–2608, Porto, Portugal, September 2001. European Union Control Association. 24

[31] A. Richards and J. How. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *Proceedings of the IEEE American Control Conference*, pages 1936–1941, Anchorage, AK, May 2002. 24

[32] A. Richards and J. How. Mixed-integer programming for control. In *Proceedings of the IEEE American Control Conference*, pages 2676–2683, Portland, OR, June 2005. 24

[33] A. Richards, J. P. How, T. Schouwenaars, and E. Feron. Plume avoidance maneuver planning using mixed integer linear programming. In *Proceedings of the AIAA Guidance, Navigation and Control Conference*, Montreal, Canada, August 2001. AIAA Paper 2001-4091. 24, 79

[34] A. Richards, T. Schouwenaars, J. P. How, and E. Feron. Spacecraft trajectory planning with avoidance constraints using mixed-integer linear programming.

Journal of Guidance, Control, and Dynamics, 25(4):755–764, July-August 2002. 24

[35] T. Schouwenaars, B. Mettler, E. Feron, and J. P. How. Hybrid architecture for full-envelope autonomous rotorcraft guidance. In *Proceedings of the American Helicopter Society Annual Forum*, Phoenix, May 2003. 24, 143

[36] T. Schouwenaars, E. Feron, and J. P. How. Hybrid model for receding horizon guidance of agile autonomous rotorcraft. In *Proceedings of the IFAC Symposium on Automatic Control in Aerospace*, St. Petersburg, Russia, June 2004. 24, 143

[37] T. Schouwenaars. *Safe Trajectory Planning of Autonomous Vehicles*. PhD thesis, Massachusetts Institute of Technology, February 2006. 24

[38] J. Bellingham, M. Tillerson, A. Richards, and J. P. How. Multi-task allocation and path planning for cooperating UAVs. In *Cooperative Control: Models, Applications and Algorithms at the Conference on Coordination, Control and Optimization*, pages 1–19, November 2001. 24

[39] J. S. Bellingham, M. Tillerson, M. Alighanbari, and J. P. How. Cooperative path planning for multiple UAVs in dynamic and uncertain environments. In *Proceedings of the IEEE Conference on Decision and Control*, pages 2816–2822, Las Vegas, NV, December 2002. 24

[40] J. S. Bellingham. Coordination and control of UAV fleets using mixed-integer linear programming. Master's thesis, Massachusetts Institute of Technology, September 2002. 24

[41] M. Alighanbari and Y. Kuwata. Coordination and control of multiple UAVs with timing constraints and loitering. In *Proceedings of the IEEE American Control Conference*, pages 5311–5316, Denver, CO, June 2003. 24

[42] J. Bellingham, A. Richards, and J. P. How. Receding horizon control of autonomous aerial vehicles. In *Proceedings of the IEEE American Control Conference*, pages 3741–3746, Anchorage, AK, May 2002. 24, 75, 76, 142

[43] Y. Kuwata and J. P. How. Three dimensional receding horizon control for UAVs. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Providence, RI, August 2004. AIAA Paper 2004-5141. 24, 75

[44] J. Bellingham, Y. Kuwata, and J. P. How. Stable receding horizon trajectory control for complex environments. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Austin, TX, August 2003. AIAA Paper 2003-5635. 24, 75

[45] Y. Kuwata and J. P. How. Stable trajectory design for highly constrained environments using receding horizon control. In *Proceedings of the IEEE American Control Conference*, pages 902–907, Boston, MA, June/July 2004. 24, 75, 142

[46] T. Schouwenaars, E. Feron, and J. P. How. Safe receding horizon path planning for autonomous vehicles. In *Proceedings of the 40th Allerton Conference on Communication, Control, and Computation*, Allerton Park, IL, October 2002. 25, 73, 74

[47] T. Schouwenaars, J. P. How, and E. Feron. Receding horizon path planning with implicit safety guarantees. In *Proceedings of the IEEE American Control Conference*, pages 5576–5581, Boston, MA, June/July 2004. 25, 73

[48] A. Richards and J. P. How. Model predictive control of vehicle maneuvers with guaranteed completion time and robust feasibility. In *Proceedings of the IEEE American Control Conference*, pages 4034–4040, Denver, CO, June 2003. 25, 73

[49] A. Richards and J. P. How. Robust variable horizon model predictive control for vehicle maneuvering. *International Journal of Robust and Nonlinear Control*, 16:333–351, February 2006. 25, 73

[50] Y. Kuwata, T. Schouwenaars, A. Richards, and J. P. How. Robust constrained receding horizon control for trajectory planning. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, San Francisco, CA, August 2005. 25, 67, 73

[51] K. Culligan, M. Valenti, Y. Kuwata, and J. P. How. Three-dimensional flight experiments using on-line mixed-integer linear programming trajectory optimization. In *Proceedings of the IEEE American Control Conference*, pages 5322–5327, New York City, NY, July 2007. 25, 28, 67, 68, 78, 82, 91, 92

[52] M. G. Earl and R. DAndrea. Iterative MILP methods for vehicle control problems. *IEEE Transactions on Robotics*, 21(6):1158–1167, December 2005. 25

[53] I. Kolmanovsky and E. G. Gilbert. Theory and computation of disturbance invariant sets for discrete-time linear systems. *Mathematical Problems in Engineering*, 4:317–367, 1998. 41

[54] J. Forrest, D. de la Nuez, and R. Lougee-Heimer. Coin-or linear program solver (CLP). Online `http://www.coin-or.org/Clp/index.html`, 2004. 50

[55] J. F. Sturm. Self-dual-minimization solver (SeDuMi). Online `http://sedumi.mcmaster.ca/`, 2006. 50

[56] J. Löfberg. YALMIP toolbox. Online `http://control.ee.ethz.ch/~joloef/wiki/pmwiki.php`, 2007. 50

[57] E. C. Kerrigan. *Robust Constraint Satisfaction: Invariant Sets and Predictive Control*. PhD thesis, University of Cambridge, November 2000. 50, 74

[58] M. Kvasnica, P. Grieder, and M. Baotić. Multi-parametric toolbox (MPT). Online `http://control.ee.ethz.ch/~mpt/`, 2004. 50

[59] K. Culligan. Online trajectory planning for UAVs using mixed integer linear programming. Master's thesis, Massachusetts Institute of Technology, September 2006. 67

[60] F. Blanchini. Set invariance in control. *Automatica*, 35:1747–1767, 1999. 74

[61] E. C. Kerrigan and J. M. Maciejowski. Invariant sets for constrained nonlinear discrete-time systems with application to feasibility in model predictive control.

In *Proceedings of the IEEE Conference on Decision and Control*, pages 4951–4956, Sydney, Australia, December 2000. 74

[62] S. V. Raković, E. C. Kerrigan, D. Q. Mayne, and K. I. Kouramas. Optimized robust control invariance for linear discrete-time systems: Theoretical foundations. *Automatica*, 43:831–841, 2007. 74

[63] S. V. Raković, E. C. Kerrigan, K. I. Kouramas, and D. Q. Mayne. Invariant approximations of the minimal robust positively invariant set. *IEEE Transactions on Automatic Control*, 50(3):406–410, March 2005. 74

[64] L. P. Gewali, S. Ntafos, and I. G. Tollis. Path planning in the presence of vertical obstacles. *IEEE Transactions on Robotics and Automation*, 6(3):331–341, June 1990. 75, 142

[65] Sun Microsystems. Developer Resources for Java Technology. Online `http://java.sun.com/`, 2008. 92

[66] Inc. ILOG. ILOG CPLEX: High-performance software for mathematical programming and optimization. Online `http://www.ilog.com/products/cplex/`, 2008. 92

[67] J. P. How, B. Bethke, A. Frank, D. Dale, and J. Vian. Real-time indoor autonomous vehicle test environment. *IEEE Control Systems Magazine*, 28(2):51–64, April 2008. 111, 114

[68] S. Bouabdallah, P. Murrieri, and R. Siegwart. Design and control of an indoor micro quadrotor. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4393–4398, New Orleans, LA, April 2004. 112, 114

[69] M. Valenti, B. Bethke, G. Fiore, J. P. How, and E. Feron. Indoor multi-vehicle flight testbed for fault detection, isolation, and recovery. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Keystone, CO, August 2006. 114

[70] Aerospace Controls Laboratory. UAV swarm health management project. Online `http://vertol.mit.edu`, 2008. 114, 130

[71] M. Valenti, B. Bethke, J. How, D. Pucci de Farias, and J. Vian. Embedding health management into mission tasking for UAV teams. In *Proceedings of the IEEE American Control Conference*, New York, NY, June 2007. 115

[72] B. Bethke and J. How. Group health management of UAV teams with applications to persistent surveillance. In *Proceedings of the IEEE American Control Conference*, 2008. 115

[73] Vicon Motion Systems. Motion capture systems from Vicon. Online `http://www.vicon.com/`, 2008. 115

[74] L. Imsland, J. A. Rossiter, B. Pluymers, and J. Suykens. Robust triple mode mpc. In *Proceedings of the IEEE American Control Conference*, pages 869–874, Minneapolis, MN, June 2006. 142

[75] D. Q. Mayne, M. M. Seron, and S. V. Raković. Robust model predictive control of constrained linear systems with bounded disturbances. *Automatica*, 41:219–224, 2005. 142