Trajectory Planning for Unmanned Vehicles using Robust Receding Horizon Control

by

Yoshiaki Kuwata
Master of Science
Massachusetts Institute of Technology, 2003

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
February 2007

Author .....
Department of Aeronautics and Astronautics
December 22, 2006

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

Certified by .....
Eric Feron
Professor

Certified by .....
Nicholas Roy
Assistant Professor

Accepted by .....
Jaime Peraire
Professor of Aeronautics and Astronautics
Chair, Committee on Graduate Students

Trajectory Planning for Unmanned Vehicles using Robust Receding Horizon Control

by

Yoshiaki Kuwata

Submitted to the Department of Aeronautics and Astronautics
on December 22, 2006, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

This thesis presents several trajectory optimization algorithms for a team of cooperating unmanned vehicles operating in an uncertain and dynamic environment. The first, designed for a single vehicle, is the Robust Safe But Knowledgeable (RSBK) algorithm, which combines several previously published approaches to recover the main advantages of each. This includes a sophisticated cost-to-go function that provides a good estimate of the path beyond the planning horizon, which is extended in this thesis to account for three dimensional motion; constraint tightening to ensure robustness to disturbances, which is extended to a more general class of disturbance rejection controllers compared to the previous work, with a new off-line design procedure; and a robust invariant set which ensures the safety of the vehicle in the event of environmental changes beyond the planning horizon. The system controlled by RSBK is proven to robustly satisfy all vehicle and environmental constraint under the action of bounded external disturbances.

Multi-vehicle teams could also be controlled using centralized RSBK, but to reduce computational effort, several distributed algorithms are presented in this thesis. The main challenge in distributing the planning is to capture the complex couplings between vehicles. A decentralized form of RSBK algorithm is developed by having each vehicle optimize over its own decision variables and then locally communicate the solutions to its neighbors. By integrating a grouping algorithm, this approach enables simultaneous computation by vehicles in the team while guaranteeing the robust feasibility of the entire fleet. The use of a short planning horizon within RSBK enables the use of a very simple initialization algorithm when compared to previous work, which is essential if the technique is to be used in dynamic and uncertain environments. Improving the level of cooperation between the vehicles is another challenge for decentralized planning, but this thesis presents a unique strategy by enabling each vehicle to optimize its own decision as well as a feasible perturbation of its neighboring vehicles' plans. The resulting cooperative form of the distributed RSBK is shown to result in solutions that sacrifice local performance if it benefits the overall team performance. This desirable performance improvement is achieved with

only a small increase in the computation and communication requirements.

These algorithms are tested and demonstrated in simulation and on two multi-vehicle testbeds using rovers and quadrotors. The experimental results demonstrate that the proposed algorithms successfully overcome the implementation challenges, such as limited onboard computation and communication resources, as well as the various sources of real-world uncertainties arising from modeling error of the vehicle dynamics, tracking error of the low-level controller, external disturbance, and sensing noise.

Thesis Supervisor: Jonathan P. How
Title: Associate Professor

Acknowledgments

I would like to thank several people who made this research possible. First, my advisor Professor Jonathan How guided me through this work with a lot of insights. It has been a great learning experience in many directions. I would also like to thank my thesis committee members Professor Eric Feron and Professor Nicholas Roy for their constructive comments from a different point of view. Arthur Richards was very helpful in going through the details of the work and provided many technical comments. The hardware experiments could not be performed without the help of Justin Teo for rovers and Mario Valenti for quadrotors. I would also like to thank former and current labmates of the Aerospace Controls Laboratory, especially Louis Breger, Mehdi Alighanbari, and Luca Bertuccelli, for going through this long ordeal together for the past five years. Special thanks to Kathryn Fischer for the administrative help. Lastly, I am very grateful to my family for their continuous support and encouragement over the years.

This research was funded under AFOSR grant # FA9550-04-1-0458. The testbeds were funded under the DURIP grant (AFOSR Grant # F49620-02-1-0216) and the Boeing Company.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

The role of the Unmanned Aerial Vehicle (UAV) has changed over the past five to ten years as the level of autonomy onboard the vehicle increases. The primary advantage of UAVs is that they can perform missions in dangerous environments, such as battle fields and devastated areas, without risking the life of the human pilots. The variety of application areas include war zones, surveillance, forest fire monitoring, disaster relief, and weather forecast.

The environment in which UAVs are deployed is becoming complex, where the vehicles are required to account for no-fly zones, adversarial forces, and many competing objectives. What makes the problem even harder is the dynamically changing situational awareness (SA): the environment is typically poorly known when the vehicles enter the region of operation; and new information becomes available as the mission progresses. Therefore, the plans must be generated online while adaptively accounting for the changes in the SA.

Current UAVs, such as Predator or Global Hawk, typically require several operators per aircraft, even when performing simple tasks [1, 2]. In the future, a less costly approach is envisaged, which allows a single operator to control multiple semi-autonomous vehicles [3, 4]. This requires a development of distributed onboard planning capabilities that can reduce the communication with the ground station and

enhance the autonomy of the UAV team. In order for the UAVs to further expand the capabilities and perform more complex tasks, recent research has focused on developing new planning and control architecture for unmanned systems [5].

This effort spans many aspects of planning problems including task assignment, path planning, and vehicle health management. However, dealing with all the aspects simultaneously is very complex, especially since the planning system must handle various types of uncertainties in the problem such as navigation error, unknown motion of a moving target, discovery of a new threat, and malfunction of a vehicle. As a result, hierarchical approaches have been widely used to decompose this complex problem into several layers of tractable problems [6–9], leading to a task allocation layer [10–13], a trajectory planning/guidance layer [14, 15], and a low-level/inner-loop control layer [16, 17], where each layer generates plans/controls that are robust to the uncertainties in the environment and in the vehicle system. Among these, the focus of this thesis is on the trajectory generation layer of this architecture. The goal of the research is to develop an algorithm that enables a fleet of vehicles to make tactical decisions autonomously and cooperatively in a distributed manner and generate trajectories to execute missions while accounting for the uncertainties in the problem.

1.1.1 Model Predictive Control/Receding Horizon Control

Trajectory planning for autonomous vehicles has been studied in many fields including robotics, aerial vehicles, undersea vehicles, and spacecraft [18–21]. The problem of recent interest is characterized as constrained optimization problem in which the vehicle has dynamic constraints such as speed bounds, input saturation, and limited flight envelope, and the environment has obstacles and no-fly zones to avoid. This can be formulated as an optimal control problem [22, 23], and numerical approaches, such as pseudospectral methods [24, 25] and *nonlinear trajectory generator* (NTG) [26, 27], have been proposed.

Model Predictive Control (MPC) or Receding Horizon Control (RHC) has recently drawn attention from a variety of fields because it can systematically handle constraints and multivariable systems [28–31]. The basic approach is to use a model of

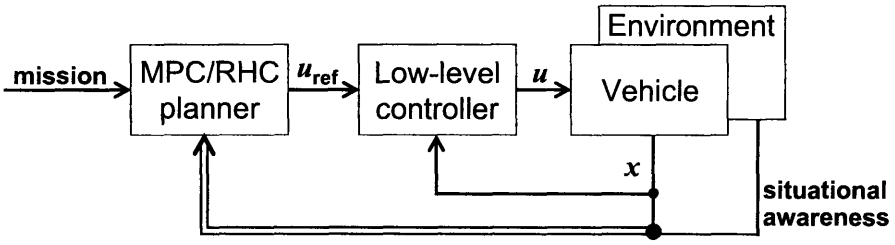


Figure 1-1: Trajectory planning using MPC/RHC. The vehicle is augmented with a low-level controller. The knowledge on the environment is handled at the planning layer.

the system to predict its future behavior and generate control inputs that satisfy the constraints and optimize the performance objectives. The optimization is repeated online as the vehicle maneuvers and new measurements about the vehicle states and the environment are obtained. Each online optimization uses the current SA, which is constantly updated based on the latest measurements and inputs from other vehicles, so the algorithm using MPC is explicitly adaptive.

Traditionally, MPC has been successfully applied to the field of process control [29, 30] where the time constants of the dynamics system are typically on the order of minutes or hours. Recent advances in the computational power has enabled the application of MPC to systems with faster dynamics, such as cars and aircraft. Because of their ability to handle the constraints, MPC/RHC are natural techniques for trajectory optimization problems [27, 32–34]. A key advantage of these optimization based controllers is that they can operate close to the constraint boundaries and obtain better performance than traditional approaches [29, 31, 35].

One challenge in applying MPC to the trajectory generation problem is to ensure that the online computations are tractable in real-time. To reduce the computational burden, RHC truncates the optimization at a finite horizon and uses a terminal penalty, also known as *cost-to-go*, which represents the rest of the trajectory [36]. Building on the roadmap method in the robotics field, our approach uses multi-resolution planning, where the trajectory planner optimizes detailed local plans over the horizon and connects it to approximate plans beyond the short planning horizon [32]. Several types of approximations have been developed to form a cost-to-go

function depending on the amount of far-field information available [8, 32, 37, 38], providing a good estimate of the remainder of the path to reach the target, even in a complicated environment. Compared to the full horizon MPC, which generates a long trajectory up to the target, these approaches avoid wasting computational resources to generate a plan in the far future, where little information is typically available and significant updates in the SA can be expected. Throughout the thesis, the planning algorithm uses this multi-resolution receding horizon strategy.

1.1.2 Robust Model Predictive Control

As stated above, the optimization based MPC/RHC controllers can operate close to the constraint boundaries [35] for a better performance. However, as a result, small disturbances could drive the system into an infeasible region, so it is important to systematically handle the uncertainties in the system. Recent research has focused on *robust MPC*, which is robust to external disturbances or inherent discrepancies between the model and the real process, and numerous techniques have been proposed in the past decade [30, 39–47].

Min-max based approaches [40, 46, 47] minimize a performance index while guaranteeing constraint satisfaction under the worst-case disturbance. The main disadvantage of this approach is that it is computationally intense, and it is not suitable for online optimization. Computationally tractable (i.e., solvable in polynomial time) approaches have been proposed using linear matrix inequalities (LMIs) [42, 45, 48]. The stability and the robustness of these LMI-based controllers have been proven using convex optimization techniques. Robust optimization [49] is also used to ensure that the solution is robust to any uncertain data realization.

The *Constraint Tightening* approaches proposed in [41, 50] are based on the idea of increasing the robustness of the controller by systematically tightening the constraints of the predicted states. The margin retained in the constraints becomes available to the MPC optimization as time progresses [51]. This margin is calculated offline to ensure room for future feedback correction in response to the disturbance. The amount of tightening has a large effect on the performance and the conservatism of the MPC

controller, and Chapter 3 presents a new offline procedure to determine the feedback policy that can tolerate much stronger disturbances and achieves much better performance in the high disturbance regime, compared to the previous work [52]. An important advantage of the constraint tightening method is that the number of decision variables and constraints in the online optimization are unchanged from the corresponding nominal MPC problem, so it is well-suited for real-time applications.

1.1.3 Multi-vehicle Control

With an increase in the mission complexity of UAVs, a tighter coordination among the vehicle fleet is becoming essential in the mission success [53, 54]. The coordination and control of multiple vehicles have many applications in the aerospace field, such as formation flying spacecraft, cooperative search and track (CSAT) mission of UAVs, and UAV rendezvous problems [8, 53–56]. To enable fleet-level cooperation, the controller must properly capture the complex interactions between the vehicles and tasks. One approach is to solve this problem globally, but centralized algorithms typically scale very poorly with the fleet size because of the computational effort involved. A natural approach to decompose the centralized problem is to let each vehicle optimize its own decision variables. The vehicles then need to communicate the solutions with each other, so that the fleet as a whole executes consistent plans.

A key challenge of decentralized control is to ensure that the distributed decision making leads to actions that are consistent with the actions of others and satisfy the coupling constraints [57, 58]. Various approaches have been investigated to address this problem, including treating the influence of other subsystems as an unknown disturbance [59], coupling penalty functions [33, 34, 60], partial grouping of computations [61], loitering options for safety guarantees [62], and dynamic programming [63, 64]. Some approaches involve iterative negotiations between subsystems [60, 65] and apply game theory to study convergence. Decentralization is further complicated when disturbances act on the subsystems, making the prediction of future behavior uncertain. The thesis uses robust MPC to tackle this problem so that the distributed planner is robust to environmental uncertainties and uncertainty

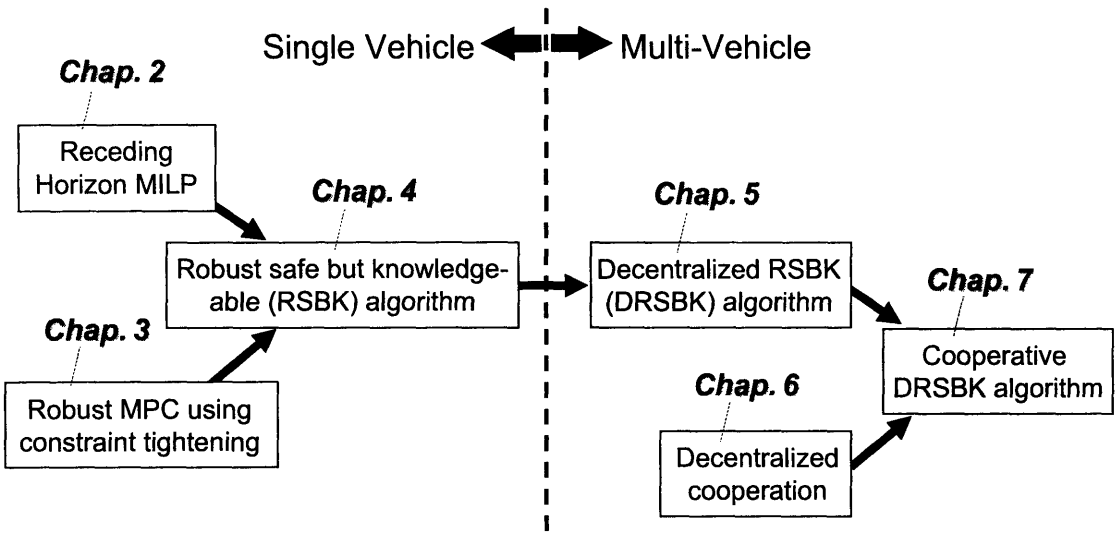


Figure 1-2: Dependency of each chapter in the thesis

in other vehicles' decisions.

1.2 Outline and Summary of Contributions

This thesis discusses both single vehicle control and multi-vehicle control using MPC. The algorithms are tested in simulation and are also demonstrated with hardware experiments. The logical dependency among the chapters are shown in Figure 1-2, and the contributions of each chapter in this thesis are summarized below.

The first three chapters deal with the control of a single vehicle.

- Chapter 2 briefly goes over the receding horizon trajectory planning strategy and the Mixed-integer Linear Programming (MILP) implementation (RH-MILP). The main contribution of this chapter is the extension of RH-MILP to three dimensions [66]. A new form of the objective function is developed that enables the vehicle to stay close to the ground but fly over the obstacles if necessary. The objective function captures the maximum rate of climb of the vehicle, and by combining it with the detailed dynamics used in MILP, the algorithm generates a kinodynamically feasible trajectory in three dimensions. Several approximations are also developed to encode the new objective function in MILP. The

overall formulation is shown to be tractable for the online planning use.

- Chapter 3 presents a robust MPC using constraint tightening. The first contribution of this chapter is the generalization of the feedback correction policy. Through the reparameterization of the feedback gain, the new algorithm is shown to represent a strictly larger class of feedback policies when compared to previous algorithms [41, 52]. The approach ensures that if the first optimization is feasible, then all future online optimization will be feasible and the system meets all the constraints under the action of disturbances.

  The second contribution of this chapter is the development of a new offline convex optimization procedure, which enables us to design a disturbance rejection controller that can tolerate much stronger disturbances. As a critical element of this procedure, the chapter derives necessary and sufficient conditions for the existence of a nonempty output constraint set, and a sufficient condition for the existence of a nonempty robust invariant set. Simulation examples show the significant performance improvements over previous approaches at high disturbance levels.

- Chapter 4 develops Robust Safe But Knowledgeable (RSBK) algorithm for trajectory optimization [67], by combining the advantages of the previous approaches [32, 68–70]. The algorithm uses the information available beyond the planning horizon to obtain a good performance with a short planning horizon; constraints are tightened in the prediction steps to save margins for future disturbance correction and robustly satisfy the constraints under the action of disturbance; and an invariant set is embedded in the online optimization, ensuring that the online optimization is always feasible even in the event of environmental changes beyond the planning horizon. The robust stability of the controller is proven by showing that the cost is monotonically decreasing.

  This chapter presents the advantages of RSBK algorithm through numerous simulation as well as hardware experiments. The algorithm is implemented on the quadrotor testbed flying in three dimensions, and its online planning

capability and the robust constraint satisfaction are demonstrated.

The last three chapters deal with multi-vehicle control using decentralized optimization.

- Chapter 5 first extends RSBK algorithm developed in Chapter 4 to the multi-vehicle scenario that includes both local and coupling constraints. The second contribution of this chapter is a distributed form of the RSBK algorithm (DRSBK) [71]. In DRSBK, each vehicle only optimizes for its own decisions by sequentially solving a subproblem of reduced size and communicating the solution to its neighbors. This results in shorter computation time compared to the centralized approach and is more suitable for real-time execution. Furthermore, DRSBK enables much simpler initialization of the algorithm than the previous work on the distributed control [52] and is well-suited for online planning. The main theoretical result in this section is that with the local planning and local communication, the algorithm is shown to guarantee the robust feasibility of the entire fleet. By integrating a grouping algorithm, the algorithm enables some vehicles to simultaneously optimize their trajectories without any conflicts, which further reduces the fleet computation time.

  Following several numerical simulations, this chapter presents results of multi-vehicle experiments, which require overcoming many implementation challenges. The onboard laptop on each rover solves the DRSBK subproblem and communicates the plan over the wireless network. The results show successful collision avoidance and obstacle avoidance using the real vehicles subject to disturbances and modeling errors, and demonstrate the distributed onboard computation capability of the DRSBK algorithm.

- Chapter 6 presents a new cooperative decentralized optimization algorithm [72]. The main contribution of this chapter is the development of the decentralized algorithm that minimizes the global performance by solving a series of "small" subproblems among the distributed vehicles, without reproducing the global

optimization problem for each vehicle. The chapter shows that the global objective is proven to monotonically decrease over iteration, and the feasibility of the entire team is maintained. Numerical simulation results show that this approach produces much better performance than the non-cooperative decentralized algorithm and is more scalable than the centralized approach.

- Chapter 7 presents a cooperative form of DRSBK algorithm for the multi-vehicle trajectory optimization, by extending the technique introduced in Chapter 6 to the MPC framework. The algorithm presented in this chapter achieves cooperative behaviors by enabling vehicles to sacrifice the local cost if it leads to the improvement of the global cost. The global cost is shown to monotonically decrease even under the action of disturbances. Finally, the hardware experiment on the multi-vehicle quadrotor testbed demonstrates the advantages of the algorithm.

Chapter 2

Three Dimensional Receding Horizon Control for UAVs

This chapter presents a receding horizon controller (RHC) that can be used to design trajectories for an aerial vehicle flying through a three dimensional terrain with obstacles and no-fly zones. To avoid exposure to threats, the paths are chosen to stay as close to the terrain as possible, but the vehicle can choose to pop-up over the obstacles if necessary. The approach is similar to the previous two-dimensional algorithms in [32, 68], which first construct a coarse cost map to provide approximate paths from a sparse set of nodes to the goal and then use Mixed-integer Linear Programming (MILP) optimization to design a detailed trajectory. The main contribution of this chapter is to extend this approach to 3D, in particular providing a new algorithm for connecting the cost map and the detailed path in the MILP. An initial guess for MILP RHC is constructed from the previous solution and is shown to reduce the solution time. Several simulation results are presented to show that the path planning algorithm yields good overall performance and is computationally tractable in a complex environment.

2.1 Introduction

With the enhancing capability of UAVs, their operation areas are being expanded to very complicated environments (e.g. urban) that have complex terrain [3]. In these environments, the vehicles can go over or go around the obstacles or no-fly zones, so path planning in three dimensions (3D) is a key technology to achieve the mission goals. In the past, vehicle guidance algorithms that avoid obstacles or other vehicles have been well studied in the areas of air traffic control, ground vehicles, and even UAVs. However, they typically assume the vehicle remains in a horizontal plane so that the path planning is two dimensional [73–75]. This chapter presents a new guidance method for vehicles flying in 3D environments to reach the target in minimum time. This method builds on the extensive literature in the fields of computational geometry and robotics on shortest path problems on 2D polygons, 3D surfaces, and 3D spaces [76–78]. Similar to previous results in [32, 68], the approach combines these shortest path algorithms with path planning techniques that use the vehicle dynamics to produce kinodynamically feasible trajectories that guide the vehicle to the goal.

The detailed trajectory optimization is conducted using Mixed-integer Linear Programming (MILP) [39, 79], which is well-suited for trajectory planning because it can directly incorporate logical constraints such as obstacle avoidance and waypoint selection and because it provides an optimization framework that can account for basic dynamic constraints such as turn limitations and maximum rate of climb. The *receding horizon* approach (RH-MILP) enables us to exploit the power of this MILP formulation in a computationally tractable algorithm [32, 68]. It solves a MILP for a detailed trajectory that only extends part of the way towards the goal. The remainder of the maneuver is represented by a cost-to-go function using path approximations. Previous work on RH-MILP presented heuristics that used straight line paths to estimate the cost-to-go from the plan's end point to the goal [32], but was limited to 2D environment. Some extensions are presented [68, 80] to compensate for the differences between the straight-line approximations in the cost-to-go calculation and the dynamically feasible paths that would be followed by the aircraft. Further extensions

are required if the vehicles are to fly close to the surface of a 3D terrain in order to avoid threats such as radars [81]. In these cases, the vertical vehicle maneuvers (*e.g.*, descend, climb up) have a significant effect on the overall trajectory, and a new cost-to-go function is needed to better estimate the future vehicle maneuvers.

This chapter extends this approach to 3D, in particular providing a new algorithm for connecting the cost map and the detailed path in the MILP. This connection is achieved by introducing a new cost-to-go function that includes an altitude penalty and accounts for the vehicle dynamics. Several simulation results are presented to show that the path planning algorithm yields good overall performance in a complex environment. Also, an algorithm to provide starting values with MILP is presented in Section 2.5, and is shown to reduce the solution time.

2.2 Algorithm Overview

The problem statement in this chapter is to design a trajectory from the current vehicle states $\boldsymbol{x}_0$ to the target location $\boldsymbol{x}_{\text{goal}}$ while minimizing some performance metric J. The vehicle dynamics are assumed to be LTI

$$\boldsymbol{x}_{k+1} = A\boldsymbol{x}_k + B\boldsymbol{u}_k, \quad \forall k \geq 0$$

and the vehicle must satisfy constraints on the states, inputs, and/or the combination of both

$$C\boldsymbol{x}_k + D\boldsymbol{u}_k \in \mathcal{Y}.$$

Figure 2-1 shows the three resolution levels used in the RH-MILP approach. In the near term, the MILP optimization solves for a detailed trajectory that extends from the current position towards the goal, but does not necessarily reach it. The line with bullets in Figure 2-1 shows this segment, which is called a planning horizon. In the far term, approximate trajectories from a set of points on the obstacles (called cost points) to the goal are solved by a graph search. The resulting information is

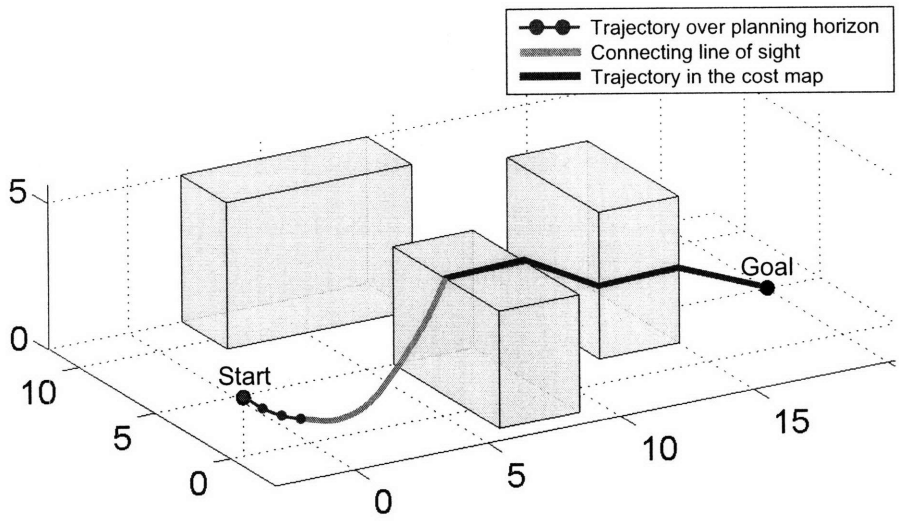


Figure 2-1: Schematic showing the three different resolution levels used in the RH-MILP approach to trajectory optimization.

stored in the cost map. The cost map is used to account for decisions beyond the planning horizon because it gives an estimate of the time to fly from each cost point to the goal. These two trajectories are then connected through the cost-to-go function in the RHC. The detailed trajectory is re-optimized online by the RHC while the vehicle executes the previous plan. The approximate trajectories are also updated online as the knowledge of the environment changes. Splitting the problem into these different levels of resolution significantly reduces the computational effort to solve for the detailed vehicle trajectory while ensuring that the future decisions are (at least approximately) taken into account.

The proposed algorithm consists of two phases: the cost map construction (Section 2.3) and the detailed trajectory optimization (Section 2.4). In the cost map construction phase, the environment is first mapped to a visibility graph consisting of nodes and arcs (Section 2.3.1). The nodes represent candidate trajectory points that the vehicle could fly through, and each arc connecting two nodes represents an approximate trajectory between them. The visibility between each pair of nodes needs to be ensured so that the arc connecting them is collision free and flyable. Section 2.3.1 presents a Linear Program (LP) that can be used to check the visibility. This new LP formulation is very flexible and can be used online for complex environments. The

next step is to compute the shortest paths from the coarse grid of nodes to the goal using Dijkstra's algorithm. The results are then stored as a cost map (Section 2.3.3). The accuracy of the path approximation depends on the node location. However, finding the exact shortest path in 3D environments is shown to be computationally intractable [82], even without the vehicle dynamics, and Section 2.3 approximates the shortest paths by introducing nodes on obstacle edges.

In the detailed trajectory optimization phase, MILP is used to formulate the overall problem. First, Section 2.4.1 presents a simple vehicle model used in the 3D trajectory planning. Section 2.4.2 presents a new cost-to-go function that is required to connect the detailed trajectory provided by MILP and the cost map produced by the graph search. Note that the limited set of nodes in the visibility graph allows the MILP to select an approximate routes from a coarse set of choices, which can significantly reduce the computation load.

2.3 Coarse Cost Map

This section presents a cost map that can be used to find approximate paths from a set of nodes to the goal. The formulation below assumes that each obstacle has a convex shape. Non-convex obstacles can be easily formed by having multiple convex obstacles intersect with each other. In 2D cases, the corners of the obstacles together with the start and the goal points form a set of nodes in the visibility graph. In the three-dimensional case, however, shortest paths rarely visit obstacle corners [77]. This chapter approximates the candidate nodes of shortest paths with obstacle corners on the ground ($z = 0$) and a middle point of each edge above ground-level. More vertices can be introduced on each obstacle edge, but the computation load both in the cost map construction phase and in the detailed trajectory design phase grows rapidly with small improvements in the accuracy [77].

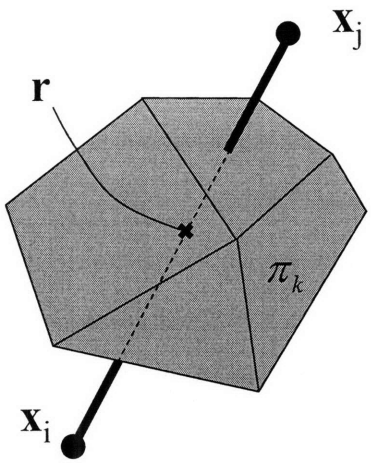


Figure 2-2: Thick line shows the arc connecting a pair of nodes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. The visibility between this pair is blocked by the obstacle π_k. The intersection point $\boldsymbol{r}$ is inside the polygon.

2.3.1 Visibility Graph in n-dimension

This section presents the visibility graph construction in an LP form. The previous RH-MILP approaches assumed that the obstacles are 2D rectangles [32, 68]. The new formulation presented in this section extends to n dimension and can handle any convex obstacles. The implementation is very simple, allowing fast computation using a commercially available LP solver such as CPLEX.

Since any obstacle can be described as a collection of convex polygons, let π_k denote the k^{th} polygon

$$\pi_k : \quad A_k \boldsymbol{r} + \boldsymbol{b}_k \leq \mathbf{0} \tag{2.1}$$

where $\boldsymbol{r} \in \mathbb{R}^n$, and the row vectors of the matrix $[A_k \mid \boldsymbol{b}_k]$ are linearly independent of each other. Polygon π_k blocks the visibility of two nodes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ if there exists a point $\boldsymbol{r}$ that satisfies (2.1) and the following conditions.

$$\boldsymbol{r} = \boldsymbol{x}_i + l\,(\boldsymbol{x}_j - \boldsymbol{x}_i) \tag{2.2}$$

$$0 \leq l \leq 1 \tag{2.3}$$

As shown in Figure 2-2, the combination of (2.2) and (2.3) ensures that the point $\boldsymbol{r}$ is on the line connecting the two nodes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, and (2.1) ensures that the point $\boldsymbol{r}$ is on or inside the polygon π_k. Given this definition, the visibility between all the nodes for all the obstacles can be determined by solving the following LP.

$$\min_{\boldsymbol{r}_{ijk}, c_{ijk}, l_{ijk}} \left(\sum_{i,j,k\,(i<j)} c_{ijk} \right) \tag{2.4}$$

subject to

$$A_k \boldsymbol{r}_{ijk} + \boldsymbol{b}_k \leq (c_{ijk} - \epsilon)\,\mathbf{1} \tag{2.5}$$

$$c_{ijk} \geq 0 \tag{2.6}$$

$$\boldsymbol{r}_{ijk} = \boldsymbol{x}_i + l_{ijk}\,(\boldsymbol{x}_j - \boldsymbol{x}_i) \tag{2.7}$$

$$0 \leq l_{ijk} \leq 1 \tag{2.8}$$

$$\forall\, i, j, k \quad (i < j)$$

where the subscripts i and j represent the nodes in the visibility graph, and the subscript k represents an obstacle. If the visibility between a node pair (i, j) is obstructed by the k^{th} obstacle, there exists a point $\boldsymbol{r}_{ijk}$ inside the obstacle such that $A_k \boldsymbol{r}_{ijk} + \boldsymbol{b}_k < 0$. This strict inequality is implemented as

$$A_k \boldsymbol{r}_{ijk} + \boldsymbol{b}_k \leq -\epsilon \mathbf{1}$$

using a small positive scalar ϵ. In such a case, (2.5) does not constrain c_{ijk}, and therefore (2.4) and (2.6) make $c_{ijk} = 0$. If the visibility is not obstructed, then (2.5) forces c_{ijk} to be positive.

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

$$c^*_{ijk} > 0, \quad \forall\, k. \tag{2.9}$$

If (2.9) is not satisfied, then at least one obstacle obstructs the visibility, as shown

in Figure 2-2. Note that the LP solution includes the visibility information on all pairs of nodes for all the obstacles, which allows for a fast incremental update of the visibility graph when the environment changes [83, 84].

2.3.2 Arc Lengths

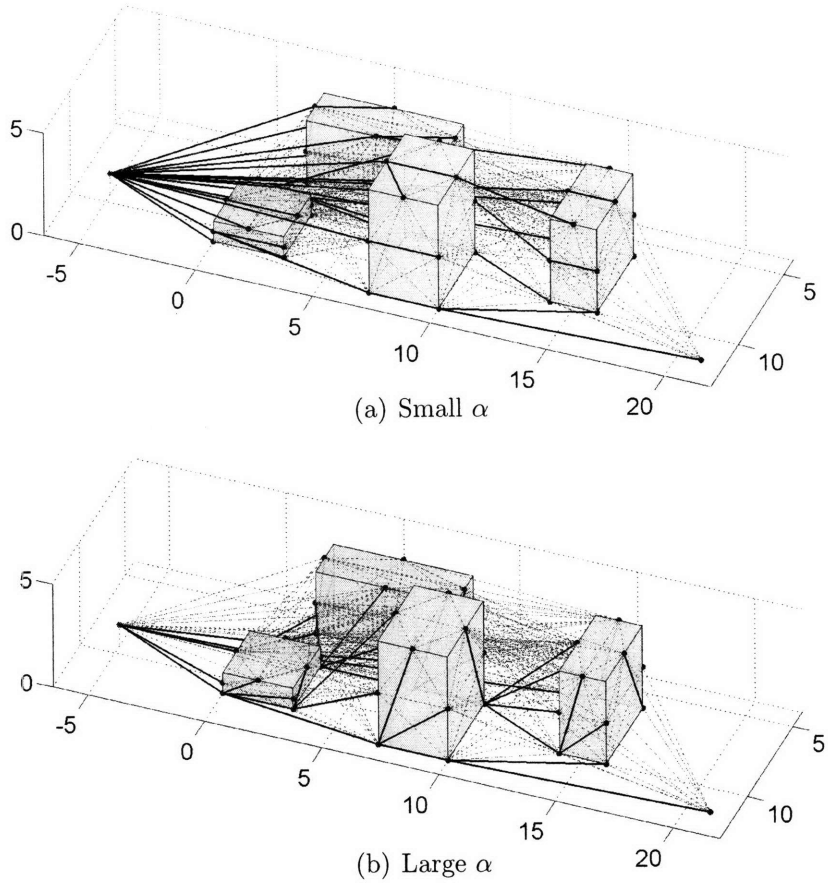
Given the visibility between the two nodes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, the next step is to calculate the arc cost D_{ij} between the two nodes, which represents the length and the threat exposure of the path connecting them. In the 3D environment, to avoid threats and radar detection, it is assumed that the vehicle would like to stay as low as possible. This objective is captured by penalizing the altitude of the path with a weight α. The focus of the cost map is to provide candidate trajectories in the far term. Thus, simple straight line trajectories are used to obtain the distance and identify the approximate threat level associated with it. The arc cost D_{ij} includes the Euclidean distance between the nodes and the path integral of the altitude along the straight line connecting the nodes.

$$D_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 \left(1 + \alpha \frac{z_i + z_j}{2}\right) \tag{2.10}$$

where $\boldsymbol{x}_i = [x_i, y_i, z_i]$.

2.3.3 Cost Map

Once the visibility graph is constructed, Dijkstra's algorithm is used to find the shortest path from each node to the goal in the visibility graph [32]. Note that the "shortest" path here is determined based on the arc cost and not necessarily the Euclidean distance. Figure 2-3 illustrates the effect of the altitude penalty α on the shortest path. The dashed lines show the visibility graph, and the thick lines show the shortest path from each node to the goal. With a small penalty on the altitude (Fig. (a)), direct connections from the goal to nodes are always shortest paths. However, with a large penalty on the altitude (Fig. (b)), the shortest paths tend to consist of arcs on the ground level.



(a) Small α

(b) Large α

Figure 2-3: Shortest path from each node (•) to the goal in the left.

The output of the Dijkstra's algorithm contains the cost C_i from each node i to the goal and the successors of each node on the way to the goal. This output is stored as a cost map and provides an approximate cost-to-go at each node in the MILP optimization, as discussed in the next section. Note that when MILP designs a path that extends towards the goal, it simply uses the nodes as a guide, and the final trajectory does not necessarily pass through these nodes.

2.4 Detailed Plan

2.4.1 Vehicle Model

The vehicle model presented in this section captures the key characteristics of the aircraft dynamics in the MILP framework [85, 86]. This is done by imposing constraints on the maximum and minimum speed, maximum turn rate, the maximum rate of climb, and the maximum rate of descent. The vehicle is assumed to have a waypoint tracking controller, and therefore its dynamics is described using a double integrator. Other vehicle models can be used in this framework to better capture more detailed vehicle dynamics such as an actuation lag [37, 87]. The linearized vehicle dynamics in discretized form can be written as

$$\begin{aligned} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{v} \end{bmatrix}_{k+1} &= A \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{v} \end{bmatrix}_{k} + B\,\boldsymbol{a}_k \\ \boldsymbol{x} &= \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad \boldsymbol{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}, \quad \boldsymbol{a} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \\ A &= \begin{bmatrix} I_3 & \Delta t \cdot I_3 \\ O_3 & I_3 \end{bmatrix}, \quad B = \begin{bmatrix} \frac{(\Delta t)^2}{2} I_3 \\ \Delta t I_3 \end{bmatrix} \end{aligned} \tag{2.11}$$

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

$$L_2(\boldsymbol{a}) \leq a_{\max} \tag{2.12}$$

$$L_2(\boldsymbol{v}) \leq v_{\max} \tag{2.13}$$

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

$$\begin{aligned} L_2(\boldsymbol{r}) &\geq \boldsymbol{r} \cdot \boldsymbol{i}_m, \qquad m = 1, \ldots, n \\ \boldsymbol{r} &= [r_x,\ r_y,\ r_z]^T \\ \boldsymbol{i}_m &= \left[\sin\phi_m \cos\theta_m,\ \sin\phi_m \sin\theta_m,\ \cos\phi_m \right]^T \end{aligned} \tag{2.14}$$

Non-convex constraints on the minimum speed

$$v_x \cos\theta_m + v_y \sin\theta_m \geq v_{\min} - 2v_{\max} b_{\text{speed},m} \qquad m = 1, \ldots, n_v \tag{2.15}$$

$$\sum_{m=1}^{n_v} b_{\text{speed},m} \geq 1 \tag{2.16}$$

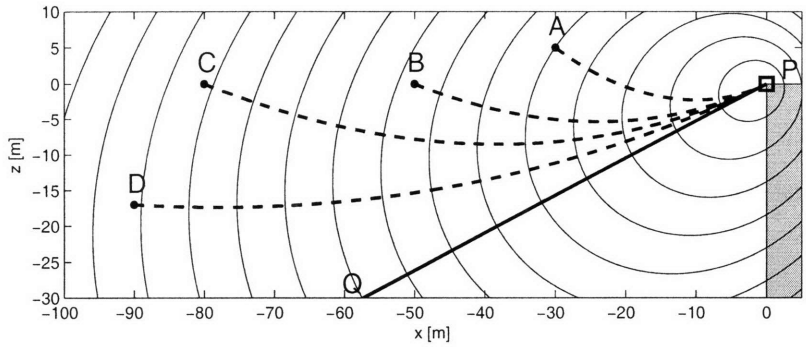
prevent the vehicle from stalling. Constraints on the maximum rate of climb and descent are implemented as

$$v_{z,\min} \leq v_z \leq v_{z,\max}. \tag{2.17}$$

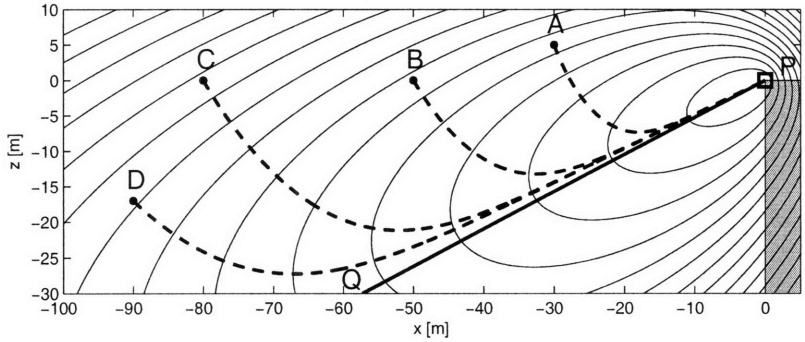
2.4.2 Cost-To-Go Function

The RHC represents the plan beyond the planning horizon by evaluating a cost-to-go function at the terminal states. The cost-to-go function in the previous work used straight lines from the terminal states to the selected cost point because it gave a good approximation of the optimal trajectory [32]. However, the terminal penalty needs to be revised to account for the change in the altitude in 3D environments. In the cost map construction phase, a line integral of the altitude along the straight line is used in (2.10) to approximate the altitude penalty in the future trajectory. In the detailed trajectory phase, the detailed altitude profile of the vehicle is obtained over the short horizon. The new cost-to-go function presented here allows us to connect these two trajectories while accounting for the altitude penalty and the vehicle dynamics.

In order to simplify the presentation, the analysis in this subsection only examines the motion in the x-z plane. The final result in Section 2.4.3 accounts for the full 3D



(a) $p = 0.25$



(b) $p = 0.75$

Figure 2-4: Contour maps of the cost-to-go function in x-z plane. Solid line represents the contour around the visible point P. Dashed lines show the steepest descent lines from four points (A, B, C, D) to the visible point. In this example, $\boldsymbol{x}_{\text{vis}} = [0, 0]^T$.

motion. Let $\boldsymbol{x}_{\text{vis}} = [x_{\text{vis}}, z_{\text{vis}}]$ denote a "visible" point that the vehicle is aiming for. Then, the cost-to-go function used in this chapter can be written as

$$F(x, z) = \sqrt{(x - x_{\text{vis}})^2 + (z - z_{\text{vis}})^2} + \alpha z - \beta(x_{\text{vis}} - x) \tag{2.18}$$
$$\alpha > 0, \ \beta > 0$$

where the first term represents the Euclidean distance between the point $[x, z]$ and the visible point $\boldsymbol{x}_{\text{vis}}$, the second and the third term separately penalize vertical and horizontal motion. In order to illustrate the effect of this cost-to-go function, Figure 2-4 shows a contour map of the cost-to-go function around the visible point P(x_{vis}, z_{vis}), which is marked with □. The dashed lines in Figure 2-4 show the steepest descent lines from four arbitrary points (A, B, C, D) to the visible point. By minimizing the cost-to-go function, the vehicle lowers its altitude to reduce the altitude penalty when

the vehicle is far from the visible point and its altitude is high. As it moves closer to the visible point, the trajectory converges to the limiting line PQ.

The second and the third term $\alpha z - \beta(x_{\text{vis}} - x)$ in (2.18) determine the elevation angle of this line PQ. It can be shown geometrically that the major axis of the ellipse in (2.18) forms an angle $\gamma_{\max}$ with x axis, where

$$\tan \gamma_{\max} = \frac{\alpha}{\beta}. \tag{2.19}$$

This angle $\gamma_{\max}$ represents the maximum path angle of the vehicle that must be embedded in the cost map. If the vehicle crosses the line PQ, it cannot avoid colliding with the gray obstacle on the right. However, by minimizing the cost-to-go function in (2.18), the vehicle trajectory will not cross the line PQ, as the plots of the steepest descent lines show.

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

$$p \equiv \alpha^2 + \beta^2 < 1 \tag{2.20}$$

otherwise, (2.18) becomes a parabola or hyperbola that has no minimum. Finally, the coefficients α and β in (2.18) can be obtained from the following equations, given the maximum path angle $\gamma_{\max}$ and a parameter p.

$$\alpha = \frac{\gamma_{\max}\sqrt{p}}{\sqrt{{\gamma_{\max}}^2 + 1}} \tag{2.21}$$

$$\beta = \frac{\sqrt{p}}{\sqrt{{\gamma_{\max}}^2 + 1}} \tag{2.22}$$

Choosing a larger p produces a flatter ellipse, and hence tighter trajectories. Figures 2-4(a) and (b) compare two contours with the same $\gamma_{\max}$ but different p. The dashed lines in the Figure 2-4(b) have tighter descent trajectories.

Note that although the cost-to-go function includes the ascending vehicle dynamics only, the vehicle dynamics over the planning horizon capture the descending dynamics. Therefore, the combination of the vehicle model in Section 2.4.1 and the cost-to-go

function produces a dynamically feasible trajectory over the planning horizon and kinodynamically feasible rate-of-climb commands beyond it.

2.4.3 RH-MILP

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

$$\boldsymbol{x}_{\text{vis}} = \sum_{i=1}^{n_{\text{cp}}} b_{\text{vis},i}\, \boldsymbol{x}_{\text{cp},i} \tag{2.23}$$

$$1 = \sum_{i=1}^{n_{\text{cp}}} b_{\text{vis},i} \tag{2.24}$$

In order to connect the detailed 3D trajectory to the selected cost point, (2.18) is extended here to 3D

$$\begin{aligned} F_i(x,y,z) = &\sqrt{(x - x_{\text{cp},i})^2 + (y - y_{\text{cp},i})^2 + (z - z_{\text{cp},i})^2} \\ &+ \alpha z - \beta \left\| \begin{bmatrix} x_{\text{cp},i} - x \\ y_{\text{cp},i} - y \end{bmatrix} \right\|_2 \qquad (i = 1, \ldots, n_{\text{cp}}) \end{aligned} \tag{2.25}$$

RHC optimizes the vehicle trajectory over a short planning horizon of N steps, executes only the first step of the control input, and starts the next optimization from the state that the vehicle will reach. Each optimization produces a detailed, but short, trajectory, which allows us to assume that the trajectory point $\boldsymbol{x}$ lies close to a vertical plane passing through a cost point $\boldsymbol{x}_{\text{cp},i}$ and the initial position $\boldsymbol{x}_k$ in the plan made at time k. In this case, the last term of (2.25) is approximated as

$$\left\| \begin{bmatrix} x_{\text{cp},i} - x \\ y_{\text{cp},i} - y \end{bmatrix} \right\|_2 \simeq \left\| \begin{bmatrix} x_{\text{cp},i} - x_k \\ y_{\text{cp},i} - y_k \end{bmatrix} \right\|_2 - \left\| \begin{bmatrix} x - x_k \\ y - y_k \end{bmatrix} \right\|_2 \tag{2.26}$$

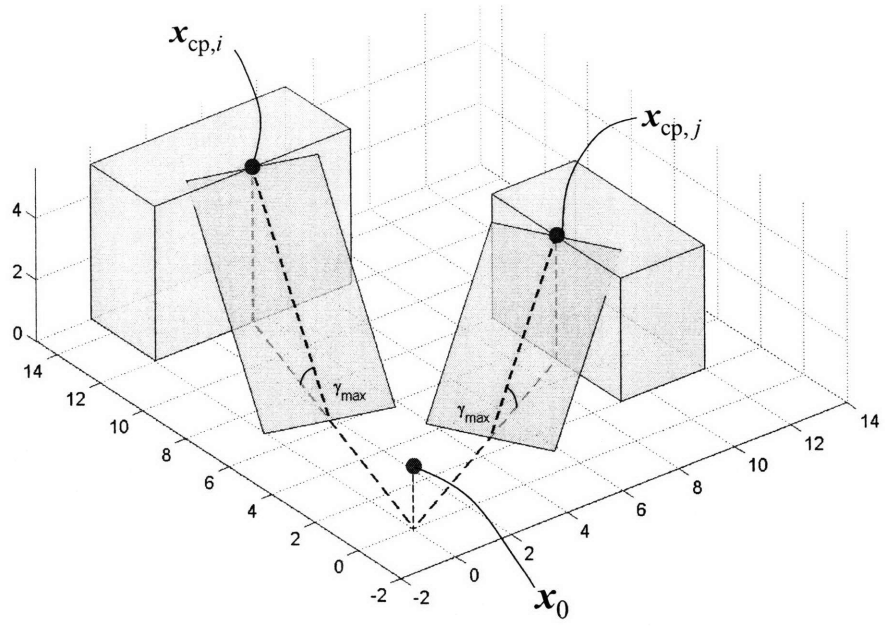


Figure 2-5: Dashed lines show path approximation by the cost-to-go function in a three dimensional environment. Each plane is formed with two axes of the contour ellipsoid.

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

$$\left\| \begin{bmatrix} x - x_k \\ y - y_k \end{bmatrix} \right\|_2 = (x - x_k)\cos\theta_i + (y - y_k)\sin\theta_i \tag{2.27}$$

$$\tan\theta_i = \frac{y_{\mathrm{cp},i} - y_k}{x_{\mathrm{cp},i} - x_k} \tag{2.28}$$

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

$\boldsymbol{x}_k$ and $\boldsymbol{x}_{\mathrm{cp},i}$. Then, (2.25) is

$$\begin{aligned} F_i(x,y,z) &\simeq \sqrt{(x-x_{\mathrm{cp},i})^2+(y-y_{\mathrm{cp},i})^2+(z-z_{\mathrm{cp},i})^2} \\ &\quad + \alpha z + \beta\left\{\left\|\begin{bmatrix} x-x_k \\ y-y_k \end{bmatrix}\right\|_2 - \left\|\begin{bmatrix} x_{\mathrm{cp},i}-x_k \\ y_{\mathrm{cp},i}-y_k \end{bmatrix}\right\|_2\right\} \\ &\simeq \sqrt{(x-x_{\mathrm{cp},i})^2+(y-y_{\mathrm{cp},i})^2+(z-z_{\mathrm{cp},i})^2} \\ &\quad + \alpha z + \beta\Big\{(x-x_k)\cos\theta_i + (y-y_k)\sin\theta_i - d_i\Big\}. \end{aligned} \tag{2.29}$$

The third term $\beta\{\cdot\}$ in (2.29) is equivalent to the third term in (2.18); it evaluates the horizontal distance from the point $\boldsymbol{x}$ to the selected cost point. For each cost point, the contour of (2.29) is ellipsoid, and its major axis makes an angle $\gamma_{\max}$ with the ground surface $z = 0$, as shown in Figure 2-5. Note that this axis is equivalent to the line PQ in Figure 2-4.

This cost-to-go function F_i must be expressed in a MILP form. The first term in (2.29) represents the two-norm of a vector, which can be approximated using a set of distributed unit vectors, as shown in (2.14). The third term $\beta\{\cdot\}$ can be obtained by minimizing βJ_{h}, where

$$J_{\mathrm{h}}(x,y) = \sum_{i=1}^{n_{\mathrm{cp}}} l_i \; - \sum_{i=1}^{n_{\mathrm{cp}}} b_{\mathrm{vis},i}\; d_i \tag{2.30}$$

with

$$l_i \geq (x-x_k)\cos\theta_i + (y-y_k)\sin\theta_i - Nv_{\max}\Delta t\,(1-b_{\mathrm{vis},i}) \tag{2.31}$$

$$l_i \geq 0 \tag{2.32}$$

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

If the i^{th} cost point is not selected, $b_{\mathrm{vis},i} = 0$, and (2.31) is relaxed. This is because the sum of the first two terms expresses the distance traveled in the direction of the i^{th} cost point, which is always smaller than the planning horizon length $Nv_{\max}\Delta t$. Minimizing J_{h} forces all the l_i's to equal zero except for the one associated with the

selected cost point ($b_{\text{vis},i} = 1$). In particular, if the i^{th} cost point is selected, then

$$\min J_{\text{h}} = \beta\Big\{(x - x_k)\cos\theta_i + (y - y_k)\sin\theta_i - d_i\Big\}$$

as required.

Each cost-to-go function (2.25) has the global minimum at the cost point $\boldsymbol{x}_{\text{cp},i}$. This can be interpreted as a potential function surrounding each cost point. The decision variable $\boldsymbol{b}_{\text{vis}}$ then makes an in-flight selection of the potential field. Path planning techniques using a potential function typically have difficulty handling local minima, but the dynamic mode switching by $\boldsymbol{b}_{\text{vis}}$ overcomes this issue.

Kinematic constraints including obstacle avoidance and the ground plane can be expressed in MILP using a binary variable $\boldsymbol{b}_{\text{obst}}$ [75]. The constraints are applied to each trajectory point over the planning horizon. To ensure that the selected cost point $\boldsymbol{x}_{\text{vis}}$ is "visible" from the terminal point $\boldsymbol{x}_{k+N}$, several sample points are placed on the line connecting these two points, and kinematic constraints are applied also to them. For each point $\boldsymbol{x} = [x, y, z]^T$ and each rectangular column shaped obstacle defined by two corners $[x_{\text{low}}, y_{\text{low}}, z_{\text{low}}]^T$ and $[x_{\text{high}}, y_{\text{high}}, z_{\text{high}}]^T$, the avoidance constraints can be expressed as

$$\left.\begin{aligned} x &\le x_{\text{low}} + M\, b_{\text{obst},1} \\ y &\le y_{\text{low}} + M\, b_{\text{obst},2} \\ z &\le z_{\text{low}} + M\, b_{\text{obst},3} \\ x &\ge x_{\text{high}} - M\, b_{\text{obst},4} \\ y &\ge y_{\text{high}} - M\, b_{\text{obst},5} \\ z &\ge z_{\text{high}} - M\, b_{\text{obst},6} \end{aligned}\right\} \tag{2.33}$$

$$z \ge 0 \tag{2.34}$$

$$\sum_{i=1}^{6} b_{\text{obst},i} \le 5 \tag{2.35}$$

where M is a large number to relax the constraints in (2.33). The logical constraint (2.35) requires at least one constraint in (2.33) be active.

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

$$\min J = \min \left\{ \sum_{j=1}^{N} \left(\|\boldsymbol{x}_{\text{vis}} - \boldsymbol{x}_{k+j}\|_2 + \alpha z_{k+j} + \beta J_{\text{h}}(x_{k+j}, y_{k+j}) \right) + \sum_{i=1}^{n_{\text{cp}}} b_{\text{vis},i} \, C_i \right\} \tag{2.36}$$

The first three terms generate a cost-to-go function to the selected cost point, as discussed in Section 2.4.2. The last term represents the cost-to-go from the selected cost point to the goal, and the values C_i's are given by the cost map, as discussed in Section 2.3.

The formulation presented in this chapter used several approximations to significantly reduce the problem size of the complex trajectory optimization. The simulation results in Section 2.6 demonstrate the validity of the approximations and show the overall MILP RHC has a good performance.

2.5 Initial Guess for MILP

In order to shorten the solution time of the MILP, an initial feasible solution can be provided with the solver. The integer feasible solution gives an upper bound on the optimal cost, which enables faster pruning of the search tree in the branch-and-bound algorithm, shortening the computation time [80]. This chapter examines 3D environments where only vertical obstacles exist. In such environments, one feasible solution is simply to fly up with its maximum acceleration.

RHC executes only the first step of the N step plan and re-optimizes from the states that will be reached. When constructing an initial guess, the decisions (e.g. visible point selection, obstacle avoidance) made in the previous solution could be used. An algorithm that constructs an initial guess from the previous solution is summarized below.

- Cost point selection

  Choose the same visible point as the one in the previous solution.

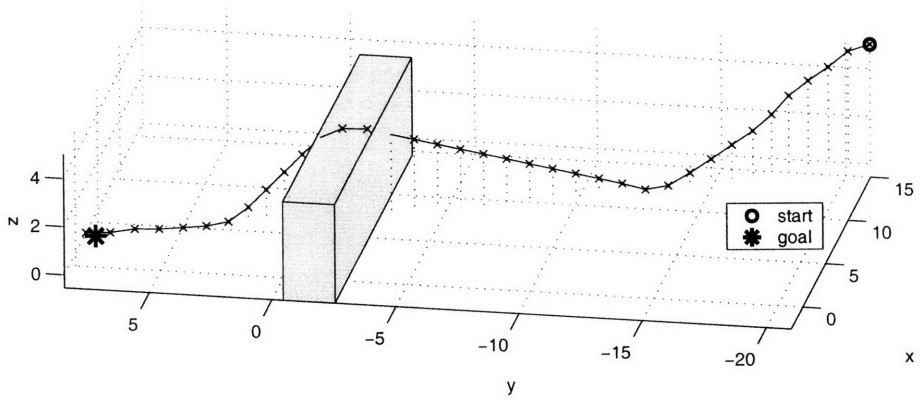


Figure 2-6: Trajectory generated by the RHC in a three dimensional environment. The vehicle starts at $\circ$, and the goal is marked with $*$.

- Input command

  For the first $(N-1)$ steps, reuse the last $(N-1)$ steps of the previous solution. For the rest, append $\boldsymbol{a} = [0, 0, a_z]^T$ where a_z is the maximum acceleration command that satisfies the constraints on the vehicle dynamics (2.11)–(2.17).

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

2.6 Simulation Results

First, a simple problem has been solved using commercially available software CPLEX 9.0 [88] on a PC (2GHz Pentium IV CPU, with 1GB RAM). Figure 2-6 shows the resulting trajectory. The following parameters are used in the simulation.

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

The start point on the right is marked with the $\circ$, and the goal is on the left. To minimize the altitude, the vehicle descends from the start point, until it reaches

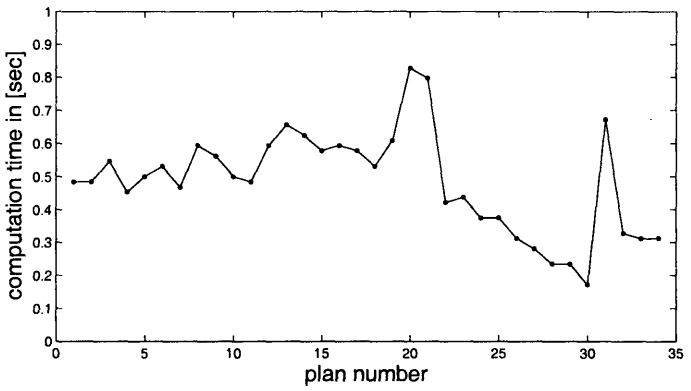
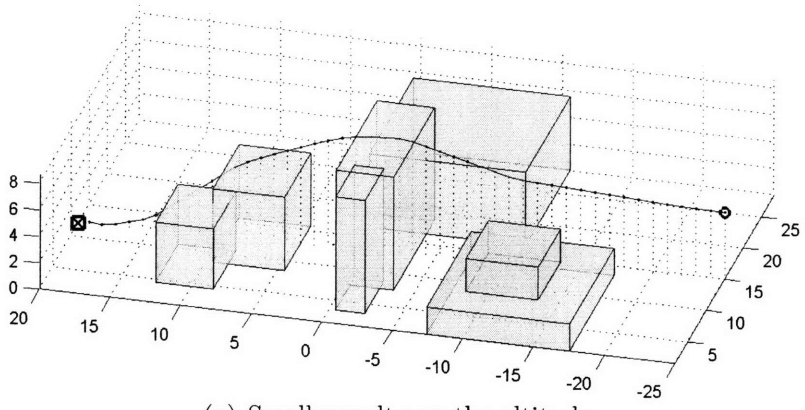


Figure 2-7: Computation time history.

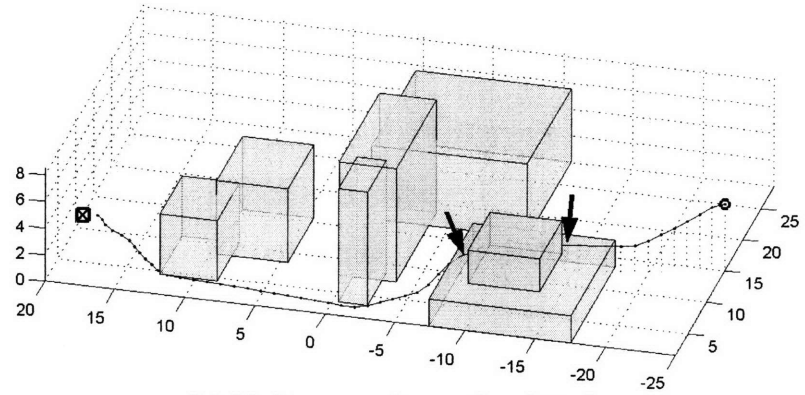
ground level. Then as it approaches the obstacle, it starts a climb-up maneuver which is triggered by the cost-to-go function (see Figure 2-4). Note that the planning horizon is only four steps in this example, and the RHC made different decisions (*e.g., descend, ascend*) while approaching the obstacle depending on the distance to the obstacle. Figure 2-7 shows the computation time for each MILP optimization.

Figure 2-8 shows trajectories in a more complicated environment. Each figure corresponds to a different penalty on the altitude. If there is only a small penalty (Fig. (a)), the vehicle flies over all of the obstacles, even including the tall ones. If projected onto the ground, the resultant trajectory is effectively a straight line connecting the start and the goal. With a larger altitude penalty (see Fig. (b)) a very different trajectory is obtained. In this case the vehicle flies around most of the obstacles at a very low altitude. However, the two-story obstacle near the start of the trajectory (lower right of the figure) is directly in the way. The vehicle decides to fly over the first-story, skirting the outside of the second story. As the altitude penalty is increased further, the vehicle goes around all the obstacles, as shown in Fig. (c). The difference between Fig. (b) and (c) is emphasized with arrows in the figures.

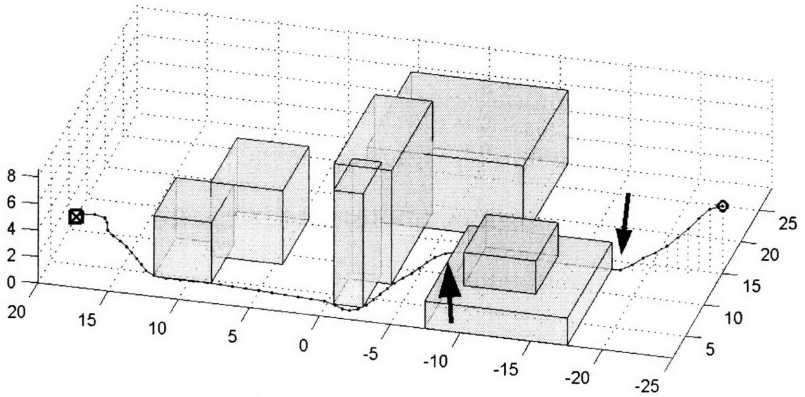
The true optimal solution is computationally intractable to obtain, but in the solutions presented here, the vehicle mostly keeps the maximum speed with the smooth trajectories, which indicates they are close to the optimal trajectory. Note that for this example the average computation time increases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



(a) Small penalty on the altitude.



(b) Medium penalty on the altitude.



(c) Large penalty on the altitude.

Figure 2-8: Trajectories generated by the RHC in a complex three dimensional environment. The vehicle starts at ∘, and the goal is marked with ⊠.

Table 2.1: Comparison of computation time (seconds)

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

Table 2.2: Reduction of the computation time (%)

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

Table 2.1 shows the CPLEX computation time in seconds for the scenarios presented in Figure 2-8. The first two columns respectively show the peak and average computation times without initial guess. The next two columns show the computation times when CPLEX is given the optimal solution as the MILP starting values. In this case, the optimal solution is first obtained and then re-used as an initial guess, which is a post-processing done only for a comparison purpose. The last two columns show the computation times when the initial guess discussed in Section 2.5 is used. Table 2.2 shows the reduction of the computation time in percentage when initial guess values are used.

There is an overall reduction of 20–28% on average if the optimal solution is provided as the MILP starting values. The initial guess in Section 2.5 produced a slightly smaller improvement in the average computation time, but can still significantly reduce the worst case computation time.

2.7 Summary

This chapter presented a trajectory planning algorithm for the vehicle flying in 3D environments with obstacles and no-fly zones. The vehicle is required to fly close to the 3D surface to avoid exposure to threats while minimizing the time of arrival at

the target. The proposed algorithm has two phases: the cost map construction and the detailed trajectory optimization. In the construction of a coarse cost map, linear programming has been applied to find the visibility graph, and the Dijkstra's algorithm is used to find the approximate shortest paths from each node to the goal. RHC designs a short but detailed trajectory using MILP while approximating the future maneuver by connecting the detailed trajectory to the coarse cost map. This is done by a new cost-to-go function which accounts for the vehicle dynamics and the altitude penalty beyond the planning horizon. The initial guess for the MILP optimization is constructed from the previous solution, which further reduces the computation load. The simulation results showed that the overall approach is computationally tractable in complex 3D environments.

Chapter 3

Robust Receding Horizon Control using Generalized Constraint Tightening

This chapter considers receding horizon control for a system that is subject to unknown but bounded disturbances. To ensure the robust constraint satisfaction of the controller, this chapter develops a new form of robust MPC using constraint tightening, where the degree of tightening is a convex function of the feedback parameters. The proposed approach is shown to be able to represent a strictly larger class of feedback policies when compared to previous algorithms. Further analytical results provide (a) necessary and sufficient conditions on the choice of feedback parameters for the existence of a nonempty output constraint set; and (b) a sufficient condition for the existence of a nonempty robust invariant set. Combined with the convex parameterization, this enables an offline linear optimization to determine the feedback policy that can tolerate much stronger disturbances while robustly satisfying the constraints. Simulation results are presented to highlight the advantages of the new control algorithm.

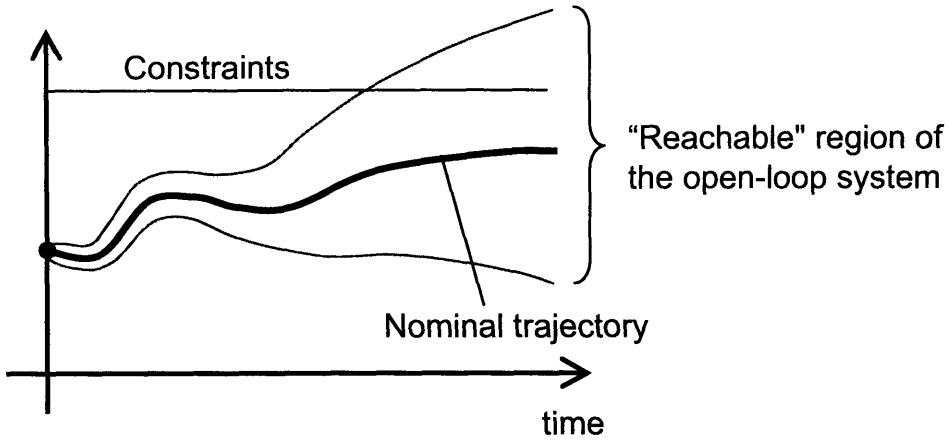
3.1 Introduction

When the system is subject to the external disturbances, the uncertainty in the system evolution grows quickly with the prediction time if there is no feedback correction, as shown in Figure 3-1(a). This so-called *open-loop prediction* is very conservative, and with a long planning horizon, it is often infeasible to ensure all possible state evolution will meet the constraints.

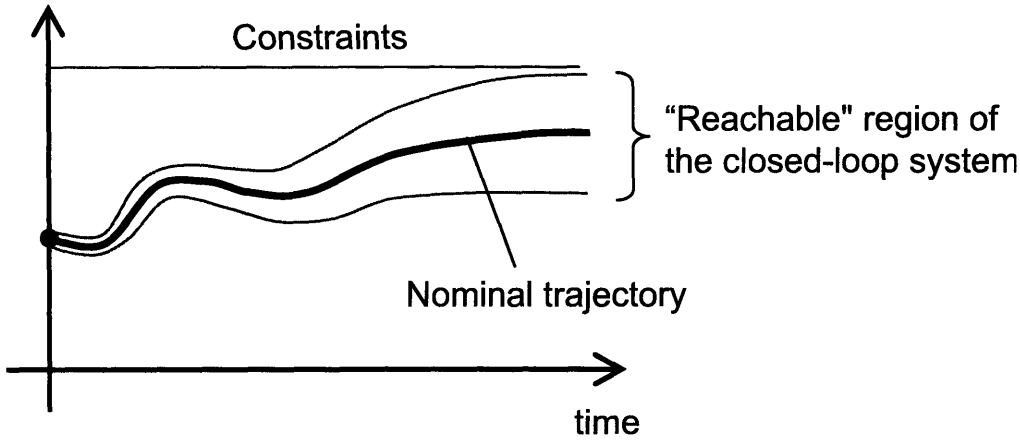
A better approach is to introduce a feedback correction controller around the nominal trajectory when predicting the future state evolution, which is called *closed-loop prediction* [31, 89]. If the disturbance is bounded, a feedback correction policy can be designed so that the closed-loop system will lie inside a bounded "reachable set" centered around the nominal trajectory. Then, in order to guarantee the robust constraint satisfaction, one needs to ensure that all trajectories of the closed-loop system satisfy the constraints, as shown in Figure 3-1(b).

The constraint tightening approach subtracts this bounded set from the constraints offline, as shown in Figure 3-1(c), so that the online optimization only needs to consider the nominal trajectory with the modified constraints. This approach is less computationally intensive than optimizing a feedback policy online [46, 47, 90], but less conservative than open-loop prediction [91, 92]. The recent extensions of constraint tightening include the use of the maximal robust control invariant admissible set as a terminal set [52], and the use of a nonlinear terminal control law and time-varying feedback correction [93], which further reduces the conservatism of the controller.

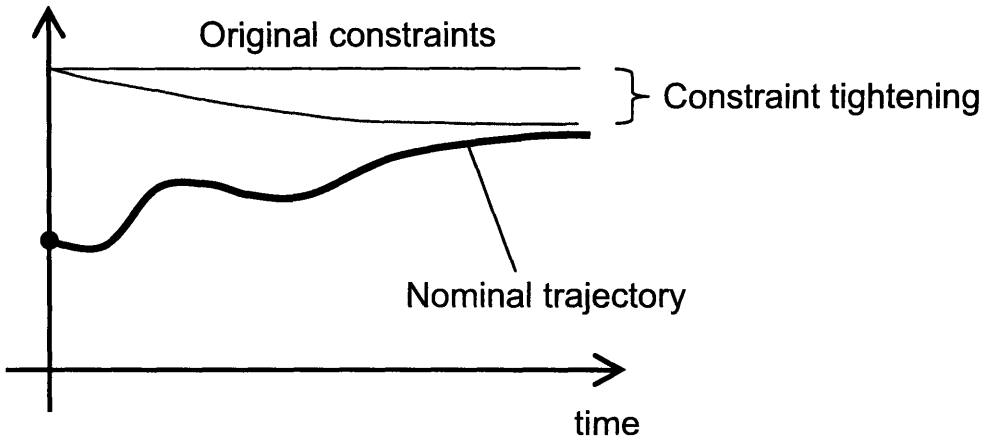
The key question for constraint tightening approaches is how to compute offline the candidate feedback policy, which in turn determines how the constraints are tightened. Recent work on MPC that optimizes the control policy online has shown that the set of feasible policies is convex if it is parameterized as *disturbance feedback* instead of state feedback [46, 90, 94]. The new method in this chapter exploits this observation to extend the constraint tightening algorithms, with the result that the offline determination of the tightened constraints can be written as an optimization over



(a) Set of possible open-loop trajectories under the action of the disturbances



(b) Set of possible closed-loop trajectories with constraints



(c) Nominal trajectory with tightened constraints

Figure 3-1: Constraint tightening approach

a convex set of disturbance feedback policies. A theorem is presented that shows all state feedback policies can be expressed as disturbance feedback, so the new method strictly subsumes existing constraint tightening work [93]. Furthermore, the determination of a robustly invariant terminal set is included in the offline optimization, by first deriving sufficient conditions for the existence of nonempty constraint sets.

This chapter is organized as follows. Following a problem statement in Section 3.3, Section 3.4 presents a generalized constraint tightening algorithm that uses a convex feedback correction controller. Section 3.5 formulates an offline optimization to obtain a controller that can handle strong disturbances and give better performance. Section 3.6 demonstrates the advantages of the proposed approach through simulations.

3.2 Notation

The Minkowski sum "$\oplus$" and the Pontryagin difference "$\sim$" of two sets $\mathcal{X}$ and $\mathcal{Y}$ are defined as follows [95].

$$\mathcal{X} \oplus \mathcal{Y} = \{z \,|\, z = x + y,\, x \in \mathcal{X},\, y \in \mathcal{Y}\}$$
$$\mathcal{X} \sim \mathcal{Y} = \{z \,|\, z + y \in \mathcal{X},\, \forall y \in \mathcal{Y}\}$$

The operation $\bigoplus_{i=0}^{n} \mathcal{X}_i = \mathcal{X}_0 \oplus \mathcal{X}_1 \oplus \cdots \oplus \mathcal{X}_n$ denotes the Minkowski summation of multiple sets. An n-dimensional p-norm ball $\mathbb{B}_p^n(\epsilon)$ with radius ϵ is defined by $\mathbb{B}_p^n(\epsilon) = \{x \,|\, \|x\|_p \leq \epsilon\}$. Unless otherwise noted, $\forall j$ implies $\forall j = 0, \ldots, N-1$ where N is a planning horizon, $\forall j^-$ implies $\forall j = 0, \ldots, N-2$, and $\forall j^+$ implies $\forall j = 0, \ldots, N$. A matrix I_n is an identity matrix of size n. $\mathbf{1}$ is a column vector of appropriate size whose elements are all 1's.

3.3 Problem Formulation

The LTI system dynamics subject to bounded disturbance and state/input constraints are

$$\boldsymbol{x}_{k+1} = A\boldsymbol{x}_k + B\boldsymbol{u}_k + \boldsymbol{w}_k \tag{3.1}$$

$$\boldsymbol{y}_k = C\boldsymbol{x}_k + D\boldsymbol{u}_k \in \mathcal{Y} \tag{3.2}$$

$$\boldsymbol{w}_k \in \mathcal{W} \tag{3.3}$$

The system (A, B) is assumed to be stabilizable and the full states are assumed to be accurately measured, although this assumption can be removed [96]. Equation (3.2) captures the constraints on the states, the inputs, or combinations of both by using the general output vector $\boldsymbol{y}_k$. The disturbance $\boldsymbol{w}_k$ is uncertain but lies in the bounded set $\mathcal{W}$, which is assumed to be known. The set $\mathcal{W}$ is also assumed to include the origin.

The overall goal is to keep the system in a feasible region using an admissible control (3.2), under the action of disturbances (3.3), while minimizing some performance criteria.

3.4 Constraint Tightening Algorithm

This section generalizes the constraint tightening robust MPC [93] using a formulation similar to the convex control parameterization in [46, 90, 94].

3.4.1 Online Optimization

At time k, the MPC controller generates a N-step control input sequence $\boldsymbol{u}_{k+j|k}$, $j = 0, \ldots, N-1$. The index j is used for the prediction steps. The conditional notation $\boldsymbol{u}_{k+j|k}$ denotes a vector $\boldsymbol{u}$ for time step $k+j$ calculated at time k. The constraint tightening algorithm uses a nominal prediction with the output constraint sets $\mathcal{Y}_j$

that is different from the original constraint set $\mathcal{Y}$ in (3.2)

$$\forall j: \quad \boldsymbol{x}_{k+j+1|k} = A\boldsymbol{x}_{k+j|k} + B\boldsymbol{u}_{k+j|k} \tag{3.4}$$

$$\boldsymbol{y}_{k+j|k} = C\boldsymbol{x}_{k+j|k} + D\boldsymbol{u}_{k+j|k} \in \mathcal{Y}_j. \tag{3.5}$$

The constraints on the initial states and the terminal states are

$$\boldsymbol{x}_{k|k} = \boldsymbol{x}_k \tag{3.6}$$

$$\boldsymbol{x}_{k+N|k} \in \mathcal{X}_{\mathrm{F}} \tag{3.7}$$

where $\mathcal{X}_{\mathrm{F}}$ is a terminal set that is defined as

$$\mathcal{X}_{\mathrm{F}} = \mathcal{R}_{\mathrm{CT}} \sim L_{N-1}\mathcal{W}. \tag{3.8}$$

The set $\mathcal{R}_{\mathrm{CT}}$ is a robust control invariant set that has the following property.

$$\exists \kappa(\boldsymbol{x}):$$

$$\forall \boldsymbol{x} \in \mathcal{R}_{\mathrm{CT}} \;\Rightarrow\; \begin{cases} A\boldsymbol{x} + B\kappa(\boldsymbol{x}) + L_{N-1}\boldsymbol{w} \in \mathcal{R}_{\mathrm{CT}}, \ \forall \boldsymbol{w} \in \mathcal{W} \\ C\boldsymbol{x} + D\kappa(\boldsymbol{x}) \in \mathcal{Y}_{N-1} \end{cases} \tag{3.9}$$

This states that once the system enters $\mathcal{R}_{\mathrm{CT}}$, then there exists an admissible control that keeps the system in $\mathcal{R}_{\mathrm{CT}}$ under the action of bounded disturbance $\boldsymbol{w}$, while satisfying all the constraints. When the control law $\kappa(\boldsymbol{x})$ is fixed, the invariant set is called Robust Positively Invariant (RPI).

The matrices $L_0, \ldots, L_{N-1}$ and the output constraint sets $\mathcal{Y}_0, \ldots, \mathcal{Y}_{N-1}$ are defined

Algorithm 3.1 MPC with Generalized Constraint Tightening

1: Assume a stabilizing terminal controller $\kappa(\boldsymbol{x})$ is given
2: Design the disturbance feedback controller P_j and L_j
3: Calculate the output constraint sets $\mathcal{Y}_j$'s and the terminal set $\mathcal{X}_{\mathrm{F}}$
4: **for** $k = 0$ to $k = \infty$ **do**
5: Take a measurement of the current states $\boldsymbol{x}_k$
6: Solve optimization subject to (3.4)–(3.7) and obtain the control inputs $\boldsymbol{u}_{k+j|k}$
7: Apply the first step of the control to the system (3.1): $\boldsymbol{u}_k = \boldsymbol{u}_{k|k}$
8: Go to the next time step
9: **end for**

by the parameters $P_1, \ldots, P_{N-1}$ using the following recursion

$$L_0 = I \tag{3.10}$$

$$L_{j+1} = AL_j + BP_{j+1} \tag{3.11}$$

$$\mathcal{Y}_0 = \mathcal{Y} \tag{3.12}$$

$$\mathcal{Y}_{j+1} = \mathcal{Y}_j \sim (CL_j + DP_{j+1})\mathcal{W}. \tag{3.13}$$

The parameters $P_1, \ldots, P_{N-1}$ are the direct feedback on the disturbances, as shown later in (3.14a), and are designed offline. Note that L_j's and $\mathcal{Y}_j$'s in (3.11) and (3.13) are linear in these parameters P_j's, which enables us to formulate an offline optimization procedure in a convex form, as presented in Section 3.5. Algorithm 3.1 shows the proposed constraint tightening MPC algorithm.

3.4.2 Robust Feasibility

Theorem 3.1. *The system* (3.1) *controlled by Algorithm 3.1 satisfies the output constraint* (3.2) *under the action of the bounded disturbance* (3.3) *for all positive* k*, if the optimization* (3.4)–(3.7) *at initial step* $k = 0$ *is feasible.*

Proof. The proof is based on a recursion which shows that for any feasible solution of the optimization at time k and disturbance realization $\boldsymbol{w}_k \in \mathcal{W}$, a feasible solution for the optimization at time $k + 1$ can be constructed.

Candidate Solution at Time $k+1$

First, assume the form of the candidate solution at time $k+1$ as

$$\hat{\boldsymbol{u}}_{k+j+1|k+1} = \boldsymbol{u}_{k+j+1|k} + P_{j+1}\boldsymbol{w}_k, \quad \forall j^- \tag{3.14a}$$

$$\hat{\boldsymbol{x}}_{k+j+1|k+1} = \boldsymbol{x}_{k+j+1|k} + L_j\boldsymbol{w}_k, \quad \forall j \tag{3.14b}$$

$$\hat{\boldsymbol{u}}_{k+N|k+1} = \kappa\big(\hat{\boldsymbol{x}}_{k+N|k+1}\big) \tag{3.14c}$$

$$\hat{\boldsymbol{x}}_{k+N+1|k+1} = A\hat{\boldsymbol{x}}_{k+N|k+1} + B\hat{\boldsymbol{u}}_{k+N|k+1} \tag{3.14d}$$

which is constructed from the solution obtained at time k. Note that the disturbance realization $\boldsymbol{w}_k$ at time k is available at time $k+1$ through $\boldsymbol{w}_k = \boldsymbol{x}_{k+1} - (A\boldsymbol{x}_k + B\boldsymbol{u}_k)$.

Initial Condition

By setting $j = 0$ in (3.14b),

$$\begin{aligned}\hat{\boldsymbol{x}}_{k+1|k+1} &= \boldsymbol{x}_{k+1|k} + \boldsymbol{w}_k \\ &= A\boldsymbol{x}_{k|k} + B\boldsymbol{u}_{k|k} + \boldsymbol{w}_k \\ &= A\boldsymbol{x}_k + B\boldsymbol{u}_k + \boldsymbol{w}_k = \boldsymbol{x}_{k+1}.\end{aligned}$$

Thus, the candidate solution satisfies the initial condition (3.6) at time k.

State Equation

Check if the candidate solution satisfies the state evolution (3.4) in the optimization at time $k+1$. The state equation for the last prediction step $j = N-1$ is obviously satisfied by the definition of $\hat{\boldsymbol{x}}_{k+N+1|k+1}$ in (3.14d). For $j = 0, \ldots, N-2$, from the definition of L_j's (3.11),

$$L_{j+1}\boldsymbol{w}_k = AL_j\boldsymbol{w}_k + BP_{j+1}\boldsymbol{w}_k$$

which holds for any $\boldsymbol{w}_k$. By adding this to the following state equation at time k

$$\boldsymbol{x}_{k+j+2|k} = A\boldsymbol{x}_{k+j+1|k} + B\boldsymbol{u}_{k+j+1|k}$$

and using (3.14a) and (3.14b), we have

$$\hat{\boldsymbol{x}}_{k+j+2|k+1} = A\hat{\boldsymbol{x}}_{k+j+1|k+1} + B\hat{\boldsymbol{u}}_{k+j+1|k+1}, \quad \forall j.$$

Output Constraints

For prediction steps $j = 0, \ldots, N-2$,

$$\begin{aligned}\hat{\boldsymbol{y}}_{k+j+1|k+1} &= C\hat{\boldsymbol{x}}_{k+j+1|k+1} + D\hat{\boldsymbol{u}}_{k+j+1|k+1} \\ &= C\boldsymbol{x}_{k+j+1|k} + D\boldsymbol{u}_{k+j+1|k} + CL_j\boldsymbol{w}_k + DP_{j+1}\boldsymbol{w}_k \\ &= \boldsymbol{y}_{k+j+1|k} + \big(CL_j + DP_{j+1}\big)\boldsymbol{w}_k.\end{aligned}$$

Note that the feasible solution at time k satisfies $\boldsymbol{y}_{k+j+1|k} \in \mathcal{Y}_{j+1}$, and the bounded disturbance is $\big(CL_j + DP_{j+1}\big)\boldsymbol{w}_k \in \big(CL_j + DP_{j+1}\big)\mathcal{W}$. Using the relation (3.13) and the following property of the Pontryagin difference

$$\boldsymbol{a} \in (\mathcal{A} \sim \mathcal{B}),\, \boldsymbol{b} \in \mathcal{B} \;\Rightarrow\; \boldsymbol{a} + \boldsymbol{b} \in \mathcal{A}$$

we have $\forall \boldsymbol{w}_k$

$$\hat{\boldsymbol{y}}_{k+j+1|k+1} \in \mathcal{Y}_j, \quad \forall j^-.$$

Terminal Constraints

Check if the terminal step of the candidate solution actually satisfies the terminal constraint $\hat{\boldsymbol{x}}_{k+N+1|k+1} \in \mathcal{X}_{\mathrm{F}}$. From (3.14b), we know

$$\hat{\boldsymbol{x}}_{k+N|k+1} = \boldsymbol{x}_{k+N|k} + L_{N-1}\boldsymbol{w}_k$$

and since

$$\boldsymbol{x}_{k+N|k} \in \mathcal{X}_{\mathrm{F}} = \mathcal{R}_{\mathrm{CT}} \sim L_{N-1}\mathcal{W}$$
$$L_{N-1}\boldsymbol{w}_k \in L_{N-1}\mathcal{W}$$

we obtain the following using the property of the Pontryagin difference.

$$\hat{\boldsymbol{x}}_{k+N|k+1} \in \mathcal{R}_{\mathrm{CT}}$$

Applying to this the invariance property (3.9) gives

$$\begin{cases} A\hat{\boldsymbol{x}}_{k+N|k+1} + B\kappa\big(\hat{\boldsymbol{x}}_{k+N|k+1}\big) + L_{N-1}\boldsymbol{w} \in \mathcal{R}_{\mathrm{CT}}, \quad \forall \boldsymbol{w} \in \mathcal{W} \\ C\hat{\boldsymbol{x}}_{k+N|k+1} + D\kappa\big(\hat{\boldsymbol{x}}_{k+N|k+1}\big) \in \mathcal{Y}_{N-1} \end{cases}$$
$$\Rightarrow \begin{cases} \hat{\boldsymbol{x}}_{k+N+1|k+1} + L_{N-1}\boldsymbol{w}_k \in \mathcal{R}_{\mathrm{CT}}, \quad \forall \boldsymbol{w}_k \in \mathcal{W} \\ C\hat{\boldsymbol{x}}_{k+N|k+1} + D\hat{\boldsymbol{u}}_{k+N|k+1} \in \mathcal{Y}_{N-1} \end{cases}$$
$$\Rightarrow \begin{cases} \hat{\boldsymbol{x}}_{k+N+1|k+1} \in \mathcal{R}_{\mathrm{CT}} \sim L_{N-1}\mathcal{W} = \mathcal{X}_{\mathrm{F}} \\ C\hat{\boldsymbol{x}}_{k+N|k+1} + D\hat{\boldsymbol{u}}_{k+N|k+1} \in \mathcal{Y}_{N-1} \end{cases}$$

Thus, the candidate solution also satisfies the terminal constraint (3.7). The last line shows that the output constraint at $j = N-1$ is satisfied.

Recursion

Assume the optimization at time k is feasible and hence the output constraint is satisfied

$$C\boldsymbol{x}_{k|k} + D\boldsymbol{u}_{k|k} \in \mathcal{Y}_0 = \mathcal{Y}.$$

Since the measured states and the control input to be implemented are written as

$$\boldsymbol{x}_{k|k} = \boldsymbol{x}_k$$

$$\boldsymbol{u}_k = \boldsymbol{u}_{k|k}$$

we have (3.2) and the real system satisfies the original output constraints $\mathcal{Y}$.

At time $k+1$, a candidate solution (3.14) can be constructed from the latest state measurement and the solution from the previous step k. As shown above, this satisfies all the constraints of the optimization at time $k+1$. Therefore, by recursion, if the optimization at initial step $k=0$ is feasible, then the optimization at all future time steps ($\forall k > 0$) remain feasible and the system satisfies the constraints. □

3.4.3 Comparison with State Feedback Parameters

This section shows that the proposed approach strictly subsumes existing constraint tightening algorithms [41, 93]. The primary difference to the previous constraint tightening formulation is the structure of the candidate solution (3.14)

$$\hat{\boldsymbol{u}}_{k+j+1|k+1} = \boldsymbol{u}_{k+j+1|k} + K_j L_j \boldsymbol{w}_k \quad \forall j^- \tag{3.15a}$$

$$\hat{\boldsymbol{x}}_{k+j+1|k+1} = \boldsymbol{x}_{k+j+1|k} + L_j \boldsymbol{w}_k \quad \forall j \tag{3.15b}$$

$$\hat{\boldsymbol{u}}_{k+N|k+1} = \kappa\big(\hat{\boldsymbol{x}}_{k+N|k+1}\big) \tag{3.15c}$$

$$\hat{\boldsymbol{x}}_{k+N+1|k+1} = A\hat{\boldsymbol{x}}_{k+N|k+1} + B\hat{\boldsymbol{u}}_{k+N|k+1}. \tag{3.15d}$$

This form of the candidate solution leads to a nonlinear relation between the feedback K_j and L_j

$$L_{j+1} = (A + BK_j)L_j \tag{3.16}$$

as opposed to the linear relation between P_j's and L_j's in (3.11). The next theorem states that the previous formulation using K_j is a strict subset of the new convex parameterization using P_j.

Theorem 3.2. *The set of feedback policies that can be expressed as* (3.14) *by choice of P_j's strictly includes the set of policies that can be expressed as* (3.15) *using K_j's.*

Proof. First, we show (3.15) is a subset of (3.14). Comparing equations (3.11) and (3.16), it can be shown that for each controller K_j, there exists an equivalent controller parameterized using P_j's, by letting

$$P_{j+1} = K_j L_j. \tag{3.17}$$

In order to show the strictness, we only need to find a counterexample where some policy represented by P_j's cannot be expressed using any K_j's. Let $k = 0$ and $j = 1, 2$ in the candidate solution (3.14a), then the first two control inputs are written using P_j's as

$$\hat{\boldsymbol{u}}_{1|1} = \boldsymbol{u}_{1|0} + P_1 \boldsymbol{w}_0, \tag{3.18}$$

$$\hat{\boldsymbol{u}}_{2|2} = \boldsymbol{u}_{2|0} + P_2 \boldsymbol{w}_0 + P_1 \boldsymbol{w}_1. \tag{3.19}$$

Using K_j's,

$$\hat{\boldsymbol{u}}_{1|1} = \boldsymbol{u}_{1|0} + K_0 L_0 \boldsymbol{w}_0, \tag{3.20}$$

$$\begin{aligned} \hat{\boldsymbol{u}}_{2|2} &= \boldsymbol{u}_{2|0} + K_1 L_1 \boldsymbol{w}_0 + K_0 L_0 \boldsymbol{w}_1 \\ &= \boldsymbol{u}_{2|0} + K_1 (A + B K_0) \boldsymbol{w}_0 + K_0 \boldsymbol{w}_1. \end{aligned} \tag{3.21}$$

In order to ensure that the same candidate control inputs $\hat{\boldsymbol{u}}_{1|1}$, $\hat{\boldsymbol{u}}_{2|2}$ are realizable, it is necessary to find K_0 and K_1 such that

$$\begin{aligned} K_0 &= P_1 \\ K_1 (A + B K_0) &= P_2. \end{aligned}$$

The first condition requires $K_0 = P_1$, so the second condition requires

$$K_1(A + BP_1) = P_2. \tag{3.22}$$

Suppose P_1 is chosen so that $(A + BP_1)$ is rank deficient. If a row of P_2 is chosen not to lie within the span of the rows of $(A + BP_1)$, then the equation (3.22) cannot be solved with any choice of K_1. Hence, the policy represented by P_j's cannot be generally expressed as a corresponding K_j. □

The significance of this result is that if the disturbance feedback policy P_j is optimized offline, the resulting controller must be at least as good as the best choice of state feedback policy K_j for the same objective. Furthermore, Section 3.5 shows the constraints (3.5) and (3.7) are linear function of P_j's, leading to an offline design procedure using convex optimization.

This result is different from [90], in which they showed a one-to-one mapping between the state feedback parameterization and the disturbance feedback parameterization. This difference comes from the structure in the constraint tightening algorithm, where the time varying correction laws K_j's and P_j's do not change from one optimization at time k to the next at time $k + 1$.

3.4.4 Remarks

Remark 3.1 (Online vs Offline)**.** It is possible to simultaneously optimize the feedback gain P and the control input $\boldsymbol{u}_{\cdot|k}$ online [90], at the expense of significant extra complexity of the online computation. The main advantage of the constraint tightening algorithms (both new and old) is that the decision space of the online optimization is the same as that of the corresponding nominal MPC. This is possible because the constraints are tightened offline using a pre-calculated feedback gain P or K. Section 3.5 discusses how to choose the feedback gain P offline.

Remark 3.2 (Control without online re-optimization)**.** It is interesting to note that adopting the candidate policy, without re-optimizing, would implement a policy of

the same form as [46] for the first N execution steps, but there are differences after time step N. In particular, the control inputs at time step $i = N + l$, $(l \geq 0)$ rejects the previous disturbances in two ways: the first $l + 1$ disturbance inputs $(w_0, \ldots, w_l)$ are rejected using $\kappa(\cdot)$ and L_{N-1}; and the last $N-1$ disturbance inputs $(w_{l+1}, \ldots, w_{l+N-1})$ are rejected using P_j. This introduces extra degrees of freedom into the constraint tightening algorithm over the approach in [46]. More details are given in Appendix 3.B.

3.5 Offline Choice of Feedback Gain

With the correction policy formulated in terms of disturbance feedback P, instead of state feedback K, one can optimize the policy using convex optimization. This section develops conditions on the feedback policy P to ensure the existence of a nonempty feasible set. Note that the conditions on the terminal set (3.9) depend on L_{N-1} which in turn depends upon the choice of P. Therefore, the choice of terminal constraint set is coupled to the choice of policy P and much of this section is devoted to identifying conditions on P for a suitable $\mathcal{R}_{\mathrm{CT}}$ to exist.

The section begins by developing conditions for a nonempty output set $\mathcal{Y}_{N-1}$, which from (3.9) is necessary for the existence of a nonempty terminal set, which in turn implies a nonempty feasible set. Then, two sufficient conditions are shown to ensure the existence of a nonempty terminal set $\mathcal{X}_{\mathrm{F}}$ satisfying (3.8) and (3.9).

In this section, the disturbance is assumed to be described as a polyhedron that contains the origin

$$\mathcal{W} = \left\{ \boldsymbol{w} \,|\, H_w \boldsymbol{w} \leq K_w \right\}$$

where the elements of the vector K_w are all non-negative. Let g_v represent the v-th vertex of this set $\mathcal{W}$ and $v \in \mathcal{V} = \{1, \ldots, n_v\}$. The output constraints (3.2) are also

assumed to be described by polyhedral constraints of the following form

$$\tilde{C}\boldsymbol{x}_k + \tilde{D}\boldsymbol{u}_k \leq \boldsymbol{y}_{\max} \tag{3.23}$$

where the elements of $\boldsymbol{y}_{\max}$ are all assumed to be non-negative. The set of polyhedral constraints can express the 1-norm, the approximate 2-norm, and the ∞-norm constraints. This is also written as a combination of linear constraints

$$\tilde{C}_r\boldsymbol{x}_k + \tilde{D}_r\boldsymbol{u}_k \leq y_{\max r}, \quad \forall r \tag{3.24}$$

where the subscript r is a row index of the polyhedral constraint (3.23).

3.5.1 Necessary Conditions – Nonempty Output Set

This subsection converts the condition for a nonempty output set $\mathcal{Y}_{N-1}$ into inequality constraints that are linear in P_j's and a set of slack variables t.

The parameter P must be designed to ensure the output constraint set $\mathcal{Y}_{N-1}$ is nonempty through (3.13). This requires

$$\begin{aligned} \mathcal{W}_{\text{all}} &\triangleq (CL_0+DP_1)\mathcal{W} \oplus \cdots \oplus (CL_{N-2}+DP_{N-1})\mathcal{W} \\ \mathcal{Y} \sim \mathcal{W}_{\text{all}} &\neq \emptyset \end{aligned} \tag{3.25}$$

From the recursive equation (3.10) and (3.11), matrices L_j's are given by

$$L_j = A^j + \sum_{l=0}^{j-1} A^{j-1-l}BP_{l+1}. \tag{3.26}$$

Using slack variables t and using (3.26), the constraint (3.25) is written as

$$\left(\tilde{C}_rA^j + \sum_{l=1}^{j} \tilde{C}_rA^{j-l}BP_l + \tilde{D}_rP_{j+1}\right) g_v \leq t_{rj}, \qquad \forall v,\ \forall r,\ \forall j^- \tag{3.27}$$

$$\exists \boldsymbol{x},\ \exists \boldsymbol{u}: \quad (\tilde{C}_r\boldsymbol{x} + \tilde{D}_r\boldsymbol{u}) + \sum_{j=0}^{N-2} t_{rj} \leq y_{\max r}, \qquad \forall r. \tag{3.28}$$

When $\kappa(\boldsymbol{x})$ is stabilizing, the invariant set has to include the origin as an admissible state. This requires (3.28) to be modified as

$$\sum_{j=0}^{N-2} t_{rj} \leq y_{\max r}, \quad \forall r. \tag{3.29}$$

Note that (3.27) and (3.29) represent a combination of linear constraints on the P_j's and hence a convex set of P_j's.

3.5.2 Sufficient Conditions

This section presents two alternative, sufficient conditions for the existence of a nonempty feasible set. Both retain the convexity of the optimization to choose P.

Sufficient Conditions based on Nilpotent P

The first condition uses a nilpotent policy. The RPI set $\mathcal{R}_{\text{CT}}$ can be made *nominally* invariant if a nilpotent policy $L_{N-1} = 0$ is used. It is sometimes not tractable to compute an RPI set for complex systems, whereas a nominal control invariant set is computationally much simpler to obtain. If nilpotency is desirable, one can impose the following linear constraint

$$L_{N-1} = A^{N-1} + \sum_{l=1}^{N-1} A^{N-1-l} B P_l = 0. \tag{3.30}$$

Then, the property (3.9) of the invariant set $\mathcal{R}_{\text{CT}}$ becomes

$$\forall \boldsymbol{x} \in \mathcal{R}_{\text{CT}} \Rightarrow \begin{cases} A\boldsymbol{x} + B\kappa(\boldsymbol{x}) \in \mathcal{R}_{\text{CT}} \\ C\boldsymbol{x} + D\kappa(\boldsymbol{x}) \in \mathcal{Y}_{N-1} \end{cases} \tag{3.31}$$

The set $\mathcal{R}_{\text{CT}}$ always exists when (3.27) and (3.29) ensure that the output constraint set $\mathcal{Y}_{N-1}$ is nonempty and includes the origin. In this case one could simply choose $\mathcal{R}_{\text{CT}}$ to be the origin. In summary, the sufficient conditions for the existence of a nonempty feasible set are (3.27), (3.29), and (3.30).

Sufficient Conditions based on mRPI set

Although the condition shown above consists of simple constraints, the nilpotency requirement could limit the class of possible controllers to be considered. In this section, an alternative set of conditions is developed that does not assume nilpotency of policy P. In this subsection, the terminal controller $\kappa(\boldsymbol{x})$ is assumed to be linear $\kappa(\boldsymbol{x}) = K_{\mathrm{f}}\,\boldsymbol{x}$, where the gain K_{f} is stabilizing and is assumed to be given. The approach uses the minimal robust positively invariant (mRPI) set, which is contained in any RPI set [95, 97]. The mRPI set $\mathcal{F}_\infty$ for $\mathcal{R}_{\mathrm{CT}}$ is written as

$$\mathcal{F}_\infty = \bigoplus_{i=0}^{\infty} (A + BK_{\mathrm{f}})^i L_{N-1}\mathcal{W}.$$

To simplify the presentation, define a set $\mathcal{X}_{N-1}$ such that

$$\mathcal{X}_{N-1} = \left\{ x \mid (C + DK_{\mathrm{f}})x \in \mathcal{Y}_{N-1} \right\}$$

Then, the following theorem gives the necessary and sufficient condition for the existence of a nonempty $\mathcal{R}_{\mathrm{CT}}$.

Theorem 3.3. *If $\mathcal{Y}_{N-1}$ includes the origin, then a necessary and sufficient condition for the existence of a nonempty $\mathcal{X}_{\mathrm{F}}$ using the class of sets $\mathcal{R}_{\mathrm{CT}}$ that are robust positively invariant under control K_{f} is*

$$\mathcal{F}_\infty \subseteq \mathcal{X}_{N-1}. \tag{3.32}$$

Proof. (*Necessity*) Assume there exists nonempty $\mathcal{X}_{\mathrm{F}}$. Then, there exists a nonempty RPI set $\mathcal{R}_{\mathrm{CT}}$ that has the property

$$\forall x \in \mathcal{R}_{\mathrm{CT}} \;\Rightarrow\; \begin{cases} (A + BK_{\mathrm{f}})x + L_{N-1}w \in \mathcal{R}_{\mathrm{CT}}, \;\; \forall w \in \mathcal{W} \\ (C + DK_{\mathrm{f}})x \in \mathcal{Y}_{N-1}. \end{cases}$$

or

$$\forall x \in \mathcal{R}_{\mathrm{CT}} \Rightarrow \begin{cases} (A+BK_{\mathrm{f}})x \in \mathcal{R}_{\mathrm{CT}} \sim L_{N-1}\mathcal{W} \\ x \in \mathcal{X}_{N-1}. \end{cases} \tag{3.33}$$

The last equation indicates

$$\mathcal{R}_{\mathrm{CT}} \subseteq \mathcal{X}_{N-1}. \tag{3.34}$$

By the definition of the mRPI set, $\mathcal{F}_\infty$ is contained in all RPI sets. Therefore,

$$\mathcal{F}_\infty \subseteq \mathcal{R}_{\mathrm{CT}} \subseteq \mathcal{X}_{N-1}. \tag{3.35}$$

(*Sufficiency*) Assume $\mathcal{F}_\infty \subseteq \mathcal{X}_{N-1}$. By the definition of $\mathcal{F}_\infty$, we have

$$(A+BK_{\mathrm{f}})\mathcal{F}_\infty \oplus L_{N-1}\mathcal{W} = \mathcal{F}_\infty.$$

Consider an arbitrary element $\boldsymbol{x} \in \mathcal{F}_\infty$. If $\mathcal{F}_\infty \subseteq \mathcal{X}_{N-1}$, then

$$\begin{cases} (A+BK_{\mathrm{f}})\boldsymbol{x} + L_{N-1}\boldsymbol{w} \in \mathcal{F}_\infty \quad \forall \boldsymbol{w} \in \mathcal{W} \\ \boldsymbol{x} \in \mathcal{X}_{N-1} \end{cases}$$

Therefore, $\mathcal{F}_\infty$ is a nonempty RPI set that has the property of $\mathcal{R}_{\mathrm{CT}}$. Then, one can set $\mathcal{R}_{\mathrm{CT}} = \mathcal{F}_\infty$ and the terminal set

$$\begin{aligned} \mathcal{X}_{\mathrm{F}} &= \mathcal{F}_\infty \sim L_{N-1}\mathcal{W} \\ &= \bigoplus_{i=1}^{\infty} (A+BK_{\mathrm{f}})^i L_{N-1}\mathcal{W} \end{aligned}$$

is nonempty. □

Generally, one cannot explicitly obtain the infinite summation of Minkowski additions. The following theorem gives a tractable formulation that serves as a sufficient

condition for (3.32) to hold.

Theorem 3.4. *If the following conditions hold*

$$(A + BK_{\mathrm{f}})^s \mathcal{F}_{s-1} \subseteq \mathbb{B}_p^n(\epsilon), \tag{3.36}$$

$$\mathcal{F}_{s-1} \oplus \frac{1}{1-\alpha}\mathbb{B}_p^n(\epsilon) \subseteq \mathcal{X}_{N-1}, \tag{3.37}$$

$$\alpha := \left\|(A + BK_{\mathrm{f}})^s\right\|_p < 1 \tag{3.38}$$

for some choice of $\epsilon > 0$ *and an integer* s*, where*

$$\mathcal{F}_{s-1} = \bigoplus_{i=0}^{s-1} (A + BK_{\mathrm{f}})^i L_{N-1}\mathcal{W},$$

then $\mathcal{F}_\infty \subseteq \mathcal{X}_{N-1}$.

Proof. Pre-multiply (3.36) by $(A + BK_{\mathrm{f}})^{(m-1)s}$ to give

$$(A + BK_{\mathrm{f}})^{ms} \mathcal{F}_{s-1} \subseteq (A + BK_{\mathrm{f}})^{(m-1)s} \mathbb{B}_p^n(\epsilon).$$

Using the property of the norm,

$$\left\|(A + BK_{\mathrm{f}})^{(m-1)s}\right\|_p \leq \left(\left\|(A + BK_{\mathrm{f}})^s\right\|_p\right)^{m-1} = \alpha^{m-1}.$$

Using the definition of the matrix norm

$$\|A\|_p = \max_{\|x\|_p \leq 1} \|Ax\|_p,$$

it can be shown that

$$\begin{aligned}(A + BK_{\mathrm{f}})^{(m-1)s} \mathbb{B}_p^n(\epsilon) &\subseteq \left\|(A + BK_{\mathrm{f}})^{(m-1)s}\right\|_p \mathbb{B}_p^n(\epsilon) \\ &\subseteq \alpha^{m-1} \mathbb{B}_p^n(\epsilon),\end{aligned}$$

so that

$$(A + BK_{\mathrm{f}})^{ms}\mathcal{F}_{s-1} \subseteq \alpha^{m-1}\mathbb{B}_p^n(\epsilon).$$

Taking the Minkowski summation over all positive m gives

$$\begin{aligned}\bigoplus_{m=1}^{\infty}(A + BK_{\mathrm{f}})^{ms}\mathcal{F}_{s-1} &\subseteq \bigoplus_{m=1}^{\infty}\alpha^{m-1}\mathbb{B}_p^n(\epsilon)\\ &= \left(\sum_{m=0}^{\infty}\alpha^m\right)\mathbb{B}_p^n(\epsilon)\\ &= \frac{1}{1-\alpha}\mathbb{B}_p^n(\epsilon).\end{aligned}$$

Thus, (3.36) implies

$$\begin{aligned}\mathcal{F}_\infty = &\bigoplus_{i=0}^{s-1}(A + BK_{\mathrm{f}})^i L_{N-1}\mathcal{W}\\ &\oplus\left((A + BK_{\mathrm{f}})^s\bigoplus_{i=0}^{s-1}(A + BK_{\mathrm{f}})^i L_{N-1}\mathcal{W}\right)\\ &\oplus\left((A + BK_{\mathrm{f}})^{2s}\bigoplus_{i=0}^{s-1}(A + BK_{\mathrm{f}})^i L_{N-1}\mathcal{W}\right)\\ &\oplus\cdots\\ &= \mathcal{F}_{s-1}\oplus\left(\bigoplus_{m=1}^{\infty}(A + BK_{\mathrm{f}})^{ms}\mathcal{F}_{s-1}\right)\\ &\subseteq \mathcal{F}_{s-1}\oplus\frac{1}{1-\alpha}\mathbb{B}_p^n(\epsilon),\end{aligned}$$

and combining with (3.37), this implies $\mathcal{F}_\infty \subseteq \mathcal{X}_{N-1}$. □

Furthermore, with a stabilizing terminal controller K_{f}, the size of the p-norm ball $\mathbb{B}_p^n(\epsilon)$ in (3.36) can be made arbitrarily small using a sufficiently large s. Similarly, the norm α in (3.38) can be also made arbitrarily small. Therefore, the conservatism in the sufficiency (3.36) and (3.37) for (3.32) can be made arbitrarily small.

The above result shows that constraining the choice of policy such that (3.36) and (3.37) hold ensures the existence of a nonempty feasible set. Note that (3.36) and

(3.37) are both convex constraints on the disturbance feedback policy P.

Using a hypercube ($p = \infty$) to bound the approximation error, the constraints (3.36) for a given s is implemented as

$$\begin{bmatrix} I_n \\ -I_n \end{bmatrix} \sum_{i=0}^{s-1} (A + BK_{\mathrm{f}})^{i+s} L_{N-1}\, g_{v_i} \leq \epsilon \mathbf{1},$$
$$\forall v_0 \in \mathcal{V},\ \forall v_1 \in \mathcal{V}, \ldots, \forall v_{s-1} \in \mathcal{V}, \tag{3.39}$$

which considers all combinations of disturbance vertices. The condition (3.37) is implemented as

$$(\tilde{C}_r + \tilde{D}_r K_{\mathrm{f}}) \left(\sum_{i=0}^{s-1} (A + BK_{\mathrm{f}})^i L_{N-1}\, g_{v_i} + \frac{1}{1-\alpha} \epsilon h_b \right) \leq y_{\max r} - \sum_{j=0}^{N-2} t_{rj}$$
$$\forall v_0 \in \mathcal{V},\ \forall v_1 \in \mathcal{V}, \ldots, \forall v_{s-1} \in \mathcal{V},\ \forall r,\ \forall b, \tag{3.40}$$

The vector h_b is a b-th vertex of the unit hypercube $\mathbb{B}_p^n(1)$. Note that these constraints (3.39)–(3.40) are linear in L_{N-1} and t_{rj}, which are linear in the decision variables P_j's.

In summary, the sufficient conditions for the existence of a nonempty feasible set using mRPI are (3.27), (3.29), and (3.39)–(3.40). Note that ϵ in these equations is a variable that the offline optimization also chooses. Otherwise, with a fixed small ϵ, it might not be feasible to find an outer approximation that is as good as ϵ for a given s.

3.5.3 Maximum Disturbance Handling

Because the differences in various robust MPC algorithm become more apparent in a high disturbance regime, one criterion for choosing P is to ensure that the controller can tolerate strong disturbances. Then, this will lead to a controller that has a large feasible region.

The proposed offline procedure optimizes over P, but it also introduces a positive

scalar parameter β that scales the disturbance level

$$\mathcal{W}(\beta) = \{w \mid H_w w \leq \beta K_w\}$$

and checks if there exists a controller P that gives nonempty constraint sets for the disturbance level β. The disturbance level β linearly scales the vertices g_v and the slack variables t_{rj}. Thus, g_v and t_{rj} in (3.27)–(3.28) and (3.39)–(3.40) are replaced by βg_v and βt_{rj}, respectively.

To maintain the convexity of the problem, new variables are introduced $\gamma := \beta^{-1}$, $\delta := \gamma\epsilon$, which replace the variables β, ϵ. Combining all the constraints (3.27), (3.29), (3.39)–(3.40) and dividing all by β, the final form of the offline optimization is written as

$$\min_{\gamma,\delta,P_l,L_{N-1},t_{ij}} \gamma \tag{3.41}$$

$$\text{s.t.} \quad \left(\tilde{C}_r A^j + \sum_{l=1}^{j} \tilde{C}_r A^{j-l} B P_l + \tilde{D}_r P_{j+1}\right) g_v \leq t_{rj}, \quad \forall v,\ \forall r,\ \forall j^- \tag{3.42}$$

$$\sum_{j=0}^{N-2} t_{rj} \leq \gamma y_{\max_r}, \qquad \forall r \tag{3.43}$$

$$\begin{bmatrix} I_n \\ -I_n \end{bmatrix} \sum_{i=0}^{s-1} (A + BK_{\mathrm{f}})^{i+s} L_{N-1}\, g_{v_i} \leq \delta \mathbf{1},$$

$$\forall v_0 \in \mathcal{V},\ \forall v_1 \in \mathcal{V}, \ldots, \forall v_{s-1} \in \mathcal{V} \tag{3.44}$$

$$(\tilde{C}_r + \tilde{D}_r K_{\mathrm{f}}) \left(\sum_{i=0}^{s-1} (A + BK_{\mathrm{f}})^i L_{N-1}\, g_{v_i} + \frac{1}{1-\alpha}\delta h_b\right) \leq \gamma y_{\max_r} - \sum_{j=0}^{N-2} t_{rj},$$

$$\forall v_0 \in \mathcal{V},\ \forall v_1 \in \mathcal{V}, \ldots, \forall v_{s-1} \in \mathcal{V},\ \forall r,\ \forall b \tag{3.45}$$

$$L_{N-1} = A^{N-1} + \sum_{l=1}^{N-1} A^{N-1-l} B P_l \tag{3.46}$$

which maximizes the disturbance level β, while ensuring the existence of a controller P_j that gives nonempty constraint sets $\mathcal{Y}_j$ and $\mathcal{X}_{\mathrm{F}}$. Note that all the constraints in this optimization are linear in the decision variables $\gamma, \delta, P_j, L_{N-1}$, and t_{ij}.

By maximizing the disturbance level, the resulting robust MPC can be applied to a system subject to a very strong disturbance, where other feedback correction controllers would make the tightened set empty and cannot even find a feasible solution to the first online optimization.

3.6 Numerical Example

This section shows how the offline procedure presented in Section 3.5 improves the controller's performance under the action of disturbances. The calculation of invariant sets, Pontryagin difference, and Minkowski sum are done using the set operation tools in Refs. [98, 99].

A simple double integrator is used here

$$x_{k+1} = Ax_k + Bu_k + w_k$$
$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix}$$

and the constraints are

$$-x_{\max} \leq x_k \leq x_{\max} = \begin{bmatrix} 10 \\ 5 \end{bmatrix}$$
$$|u_k| \leq u_{\max} = 4.$$

The bounded disturbance w is described by

$$H_w = \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix}, \quad K_w = \begin{bmatrix} 0.3 \\ 0.3 \\ 1 \\ 1 \end{bmatrix}.$$

A stabilizing controller is used as a terminal controller $K_{\mathrm{f}} = [-1.46, -1.71]$, and the planning horizon is $N = 5$. Seven different disturbance rejection controllers are

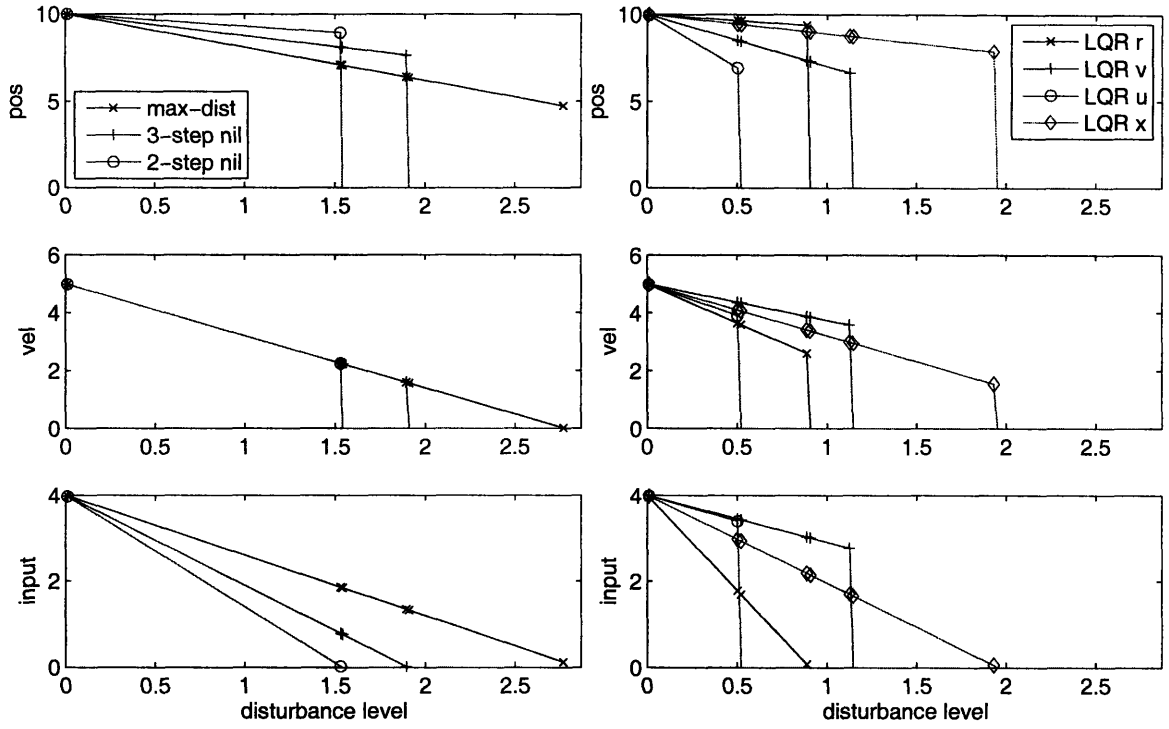
compared, each using a different policy P/K to determine the constraint tightening:

1. 2-step nilpotent controller ($L_2 = 0$) [`2-step nil`]
2. 3-step nilpotent controller ($L_3 = 0$) [`3-step nil`]
3. Controller obtained by optimization (3.41)–(3.46), with $s = 3$ and $\alpha = 0.36$. [`max-dist`]
4. LQR with $Q = \text{diag}(100, 1), R = 1$ [`LQR` r]
5. LQR with $Q = \text{diag}(1, 100), R = 1$ [`LQR` v]
6. LQR with $Q = \text{diag}(1, 1), R = 100$ [`LQR` u]
7. LQR with $Q = \text{diag}(100, 100), R = 1$ [`LQR` x]

In this example, the offline optimization (3.41)–(3.46) returned $\epsilon = 0$. From (3.36) and the definition of $\mathcal{F}_{s-1}$, this indicates that $L_{N-1} = L_4 = 0$ and the policy turns out to be nilpotent, although no such requirement was imposed.

3.6.1 Feasible Set Comparison

Figure 3-2(a) shows the limits on position, velocity and control enforced by the constraint set for the terminal step $\mathcal{Y}_{N-1}$ as a function of the disturbance level β. As the disturbance level increases, the constraints are tightened linearly. When any one of the limits falls below zero, the set $\mathcal{Y}_{N-1}$ becomes empty and the feasible set is therefore empty. Figure 3-2(b) shows the same sets $\mathcal{Y}_{N-1}$ generated by the LQR policies. Note that some controllers give empty terminal sets before any of the constraints in $\mathcal{Y}_{N-1}$ hits zero. This is because the other RPI constraint $(A + BK_{\text{f}})\boldsymbol{x} \in \mathcal{R}_{\text{CT}} \sim L_{N-1}\mathcal{W}$ becomes empty with a non-zero L_{N-1}. Observe that the `max-dist` controller can handle much larger disturbances than the others (40% stronger compared to the best of all the others), which is to be expected as that controller is designed to handle the greatest disturbance possible.

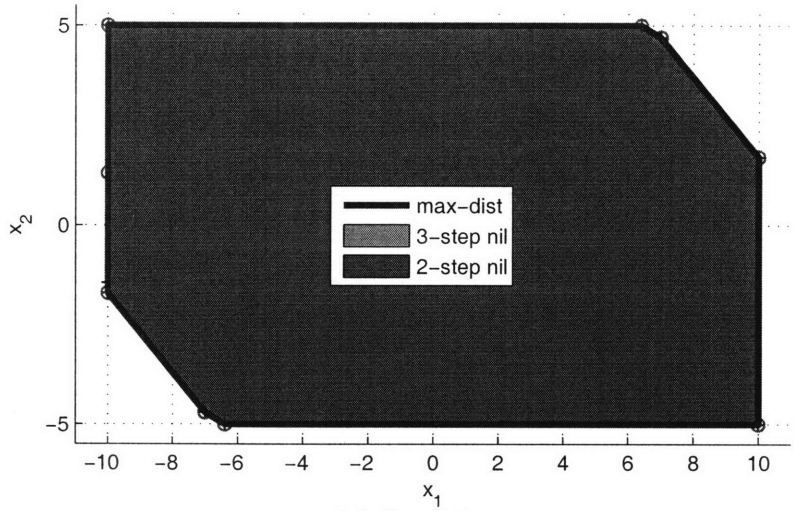


(a) Nilpotent controllers

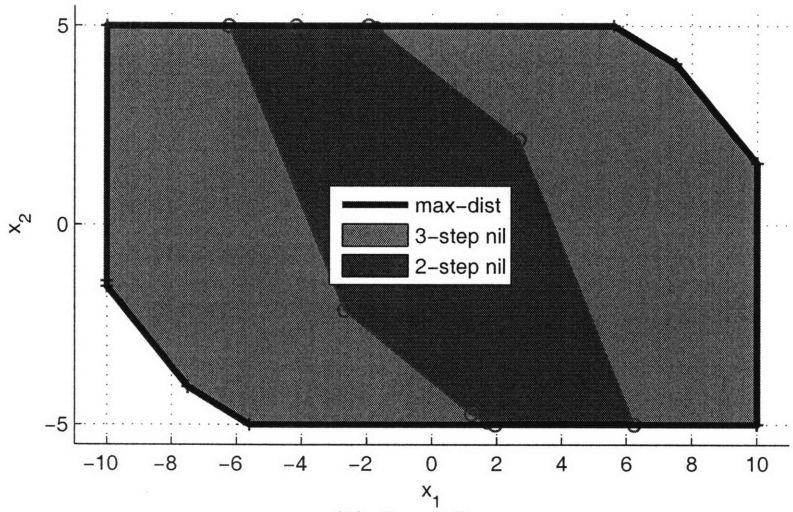
(b) LQR controllers

Figure 3-2: Terminal constraint $\mathcal{Y}_{N-1}$ as a function of disturbance level β. The controller `max-dist` keeps the constraint set $\mathcal{Y}_{N-1}$ nonempty at a disturbance level much higher than the others.

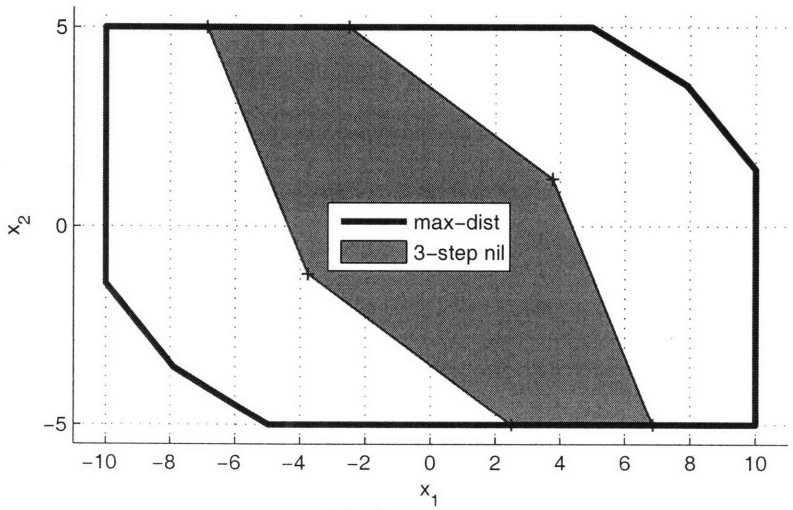
Figure 3-3 shows a set of initial states from which a controller has a feasible solution, for the three controllers `2-step nil`, `3-step nil`, and `max-dist`. As shown in Figure 3-3(b), when $\beta = 1.5$, the controllers `3-step nil` and `max-dist` have a larger set of feasible initial states compared to the controller `2-step nil`. The difference between the controllers `3-step nil` and `max-dist` is not apparent at this disturbance level. When $\beta = 1.89$, the controller `max-dist` has a feasible region much larger than the controller `3-step nil`, as shown in Figure 3-3(c). The larger set means the system has more room to operate, directly affecting the performance of the controller, as shown in the next subsection.



(a) $\beta = 1.0$



(b) $\beta = 1.5$



(c) $\beta = 1.89$

Figure 3-3: A set of states for which a feasible solution exists.

Table 3.1: Performance Comparison

	disturbance level β		
	1.0	1.5	1.89
`2-step nil`	1.017	1.792	—
`3-step nil`	1.003	1.031	1.958
`Max-dist`	**1**	**1.014**	**1.031**
`LQR` r	—	—	—
`LQR` v	0.998	—	—
`LQR` u	—	—	—
`LQR` x	1.010	1.034	1.569

3.6.2 Performance Comparison

Table 3.1 shows the effect of the disturbance level on the performance to minimize. If the controller cannot handle a given disturbance level because the constraint set $\mathcal{Y}_{N-1}$ or $\mathcal{R}_{\mathrm{CT}}$ are empty, the entry is marked with "—". The performance evaluation is quadratic

$$J = \sum_{k=0}^{k_{\max}} \left(\|x_k - x_{\mathrm{r}}\|_Q + \|u_k - u_{\mathrm{r}}\|_R \right)$$

with the following values

$$x_{\mathrm{r}} = \begin{bmatrix} 8 \\ 0 \end{bmatrix}, \qquad u_{\mathrm{r}} = 0, \qquad Q = \begin{bmatrix} 100 & 0 \\ 0 & 0 \end{bmatrix}, \qquad R = 0.$$

The random disturbance sequence is generated from the same seed for each controller. The simulation step of $k_{\max} = 8$ is used. The table lists the normalized values. The case with $\beta = 1$ shows that the choice of the controller P_j's has little effect on the performance when the system can stay away from constraints because the disturbances are weak. However, the difference becomes significant with stronger disturbances, when the constraints tend to be active and limit the performance. In such a case, the proposed controller, which has a larger feasibility region, gives the best performance (35% better compared to the best of all the others).

3.7 Summary

This chapter presented a new robust MPC algorithm using constraint tightening. The correction is described as a direct feedback of the disturbance, and a class of policies represented by the new parameterization is shown to be a strict superset of the class of policies represented by previous constraint tightening algorithms. An offline linear optimization is used to compute a feedback gain that can tolerate much stronger disturbances than other prefixed feedback gains, giving a large operating region. The simulation results show that this proposed controller leads to both a larger feasible set and performance improvements under the action of strong disturbances.

3.A RPI Set of Constraint Tightening Algorithm

Much of the work on robust MPC [41, 90, 100] assumes a fixed state feedback control beyond the horizon N. When the terminal controller $\kappa(\boldsymbol{x})$ is chosen to be a fixed linear feedback controller K_{f}, every RPI set $\mathcal{R}$ of the system (3.1)–(3.3) has the property

$$\forall x \in \mathcal{R} \;\Rightarrow\; \begin{cases} (A+BK_{\mathrm{f}})x + w \in \mathcal{R}, & \forall w \in \mathcal{W} \\ (C+DK_{\mathrm{f}})x \in \mathcal{Y}. \end{cases}$$

The terminal set of constraint tightening algorithm (3.9) uses a slightly different RPI set

$$\forall x \in \mathcal{R}_{\mathrm{CT}} \;\Rightarrow\; \begin{cases} (A+BK_{\mathrm{f}})x + L_{N-1}w \in \mathcal{R}_{\mathrm{CT}}, \; \forall w \in \mathcal{W} \\ (C+DK_{\mathrm{f}})x \in \mathcal{Y}_{N-1}. \end{cases}$$

There is a significant difference between the standard RPI set and the invariant set $\mathcal{R}_{\mathrm{CT}}$ used in constraint tightening algorithm. A simple example below shows that under the same terminal controller K_{f}, there exists a case where $\mathcal{R}$ is empty, but $\mathcal{R}_{\mathrm{CT}}$ is not.

Example: Consider the following 1-state, 1-input, 1-disturbance system with a constraint only on the state.

$$\begin{aligned} x_{k+1} &= x_k + u_k + w_k, \\ |x_k| &\le 1, \\ |w_k| &\le w_{\max}. \end{aligned}$$

Using a stabilizing terminal controller $K_{\mathrm{f}} = -0.618$, the maximal robust positively invariant set $\mathcal{R}$ is

$$\forall x \in \mathcal{R} \;\Rightarrow\; \begin{cases} 0.382x + w \in \mathcal{R}, & \forall |w| \le w_{\max} \\ |x| \le 1. \end{cases}$$

When $w_{\max} > 0.618$, this set is empty. Otherwise, $\mathcal{R} = \{x \mid |x| \leq 1\}$.

Using the same stabilizing controller $K_{\mathrm{f}} = -0.618$, the set $\mathcal{R}_{\mathrm{CT}}$ for the constraint tightening algorithm is

$$\forall x \in \mathcal{R}_{\mathrm{CT}} \quad \Rightarrow \quad \begin{cases} 0.382x \in \mathcal{R}_{\mathrm{CT}}, & \forall |w| \leq w_{\max} \\ |x| \leq 1 - w_{\max} \end{cases}$$

if P_j's are chosen such that the policy is one-step nilpotent (i.e., $L_1 = 0$). This set is empty when $w_{\max} > 1$. Otherwise, $\mathcal{R}_{\mathrm{CT}} = \{x \mid |x| \leq 1 - w_{\max}\}$.

3.B Comparison with Feedback Policy Optimization

3.B.1 Feedback Policy Optimization

Feedback Policy Optimization (FPO) [89, 90, 101] affinely parameterizes the control using a matrix $\mathbf{M}$ and a vector $\mathbf{v}$. The affine control policy is written as

$$u_i = \sum_{j=0}^{i-1} M_{i,j} w_j + v_i.$$

The online optimization solves for a pair of matrix M and vector v, so the number of decision variables is $O(N^2)$. This number of decision variables in $M_{i,j}$ can be reduced from $O(N^2)$ to $O(N)$ by using the following parameterization (the same as Eq. (7.38) of [46]).

$$\mathbf{u} = \mathbf{M}\mathbf{w} + \mathbf{v}, \qquad \mathbf{M} = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ M_{1,0} & 0 & 0 & \dots & 0 \\ M_{2,0} & M_{2,1} & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ M_{N-1,0} & M_{N-1,1} & \dots & M_{N-1,N-2} & 0 \end{bmatrix} \tag{3.47}$$

$$\mathbf{u} = \mathbf{P}\mathbf{w} + \mathbf{v}, \qquad \mathbf{P} = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ P_1 & 0 & 0 & \dots & 0 \\ P_2 & P_1 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ P_{N-1} & P_{N-2} & \dots & P_1 & 0 \end{bmatrix} \tag{3.48}$$

By comparison,

$$\begin{aligned} M_{i,j} &= M_{i+1,j+1} \qquad (1 \leq i \leq N-1,\ 0 \leq j < i) \\ P_{i-j} &= M_{i,j} \end{aligned} \tag{3.49}$$

The structure of matrix P in (3.48) states that the time varying control law (time varying with respect to the prediction step) does not change over the execution time step. The constraint tightening (CT) algorithm assumes that the feasible candidate control law (3.14) does not change from time k to $k+1$, which has the same notion as the structure in (3.48).

It can be shown that the two approaches (FPO and CT) are identical if

- the terminal set is given (the objective is to simply reach the target set, which does not need to be invariant);
- the disturbance feedback $M_{i,j}$ has a block Toeplitz structure shown in (3.49); and
- $M_{i,j}$ is precalculated and fixed in the optimization (i.e., the constraint contraction δc in [89] is obtained offline and is not a decision variable in the optimization)

However, the two approaches are different for the infinite horizon problem or the typical UAV trajectory generation problems that require a long plan. The main difference comes from the robust invariant set in which each finite horizon plan terminates. For brevity, the current time k is assumed to be 0. When the optimization is performed over N steps of control input, the decision variables are u_i $(i = 0, \dots, N-1)$ and x_i $(i = 0, \dots, N)$.

3.B.2 Control Inputs Without Re-optimization

If no re-optimization is performed, the two approaches implement the following control inputs.

CT (using the candidate solution):

$$u_{i|i} = u_{i|0} + \sum_{j=0}^{i-1} P_{i-j} w_j \qquad (i \leq N-1) \quad (3.50)$$

$$u_{N+l|N+l} = K_{\mathrm{f}}(A+BK_{\mathrm{f}})^{l}\hat{x}_N \; + \sum_{j=0}^{l} K_{\mathrm{f}}(A+BK_{\mathrm{f}})^{l-j} L_{N-1} w_j \; + \sum_{j=1}^{N-1} P_{N-j} w_{l+j}$$
$$(N \leq i = N+l). \quad (3.51)$$

After time N, the control inputs at time step $i = N + l$ rejects the previous disturbances in two ways: the first $l+1$ disturbance inputs $(w_0, \ldots, w_l)$ are rejected using K_{f} and L_{N-1}; and the last $N-1$ disturbance inputs $(w_{l+1}, \ldots, w_{l+N-1})$ are rejected using P_{N-j}.

FPO:

$$u_i = v_i + \sum_{j=0}^{i-1} M_{i,j} w_j \qquad (i \leq N-1) \quad (3.52)$$

$$u_{N+l} = K_{\mathrm{f}}(A+BK_{\mathrm{f}})^{l}\hat{x}_N \; + K_{\mathrm{f}}(A+BK_{\mathrm{f}})^{l} \sum_{j=0}^{N-1} \Big(A^{N-1-j} + \sum_{m=j+1}^{N-1} A^{N-1-m} B M_{m,j} \Big) w_j$$
$$+ \sum_{j=1}^{l} K_{\mathrm{f}}(A+BK_{\mathrm{f}})^{l-j} w_{N-1+j} \qquad (N \leq i = N+l) \quad (3.53)$$

After time N, the control inputs at time step $i = N + l$ rejects the previous disturbances in two ways: the first N disturbance inputs $(w_0, \ldots, w_{N-1})$ are rejected using K_{f} and $M_{m,j}$; and the last l disturbance inputs $(w_N, \ldots, w_{N+l-1})$ are rejected using only K_{f}.

Table 3.2: Coefficient of each term in the parameterized u

param	CT	FPO
$\hat{x}_N$	$K_{\mathrm{f}}(A+BK_{\mathrm{f}})^{l}$	$K_{\mathrm{f}}(A+BK_{\mathrm{f}})^{l}$
w_0	$K_{\mathrm{f}}(A+BK_{\mathrm{f}})^{l}\left(A^{N-1}+\sum_{m=1}^{N-1}A^{N-1-m}BP_m\right)$	$K_{\mathrm{f}}(A+BK_{\mathrm{f}})^{l}\left(A^{N-1}+\sum_{m=1}^{N-1}A^{N-1-m}BM_{m,0}\right)$
w_1	$K_{\mathrm{f}}(A+BK_{\mathrm{f}})^{l-1}\left(A^{N-1}+\sum_{m=1}^{N-1}A^{N-1-m}BP_m\right)$	$K_{\mathrm{f}}(A+BK_{\mathrm{f}})^{l}\left(A^{N-2}+\sum_{m=2}^{N-1}A^{N-1-m}BM_{m,1}\right)$
$\vdots$	$\vdots$	$\vdots$
w_l	$K_{\mathrm{f}}(A+BK_{\mathrm{f}})^{0}\left(A^{N-1}+\sum_{m=1}^{N-1}A^{N-1-m}BP_m\right)$	$K_{\mathrm{f}}(A+BK_{\mathrm{f}})^{l}\left(A^{N-(l+1)}+\sum_{m=l+1}^{N-1}A^{N-1-m}BM_{m,l}\right)$
w_{l+1}	P_{N-1}	$K_{\mathrm{f}}(A+BK_{\mathrm{f}})^{l}\left(A^{N-(l+2)}+\sum_{m=l+2}^{N-1}A^{N-1-m}BM_{m,l+1}\right)$
$\vdots$	$\vdots$	$\vdots$
w_{N-2}	P_{l+2}	$K_{\mathrm{f}}(A+BK_{\mathrm{f}})^{l}(A+BM_{N-1,N-2})$
w_{N-1}	P_{l+1}	$K_{\mathrm{f}}(A+BK_{\mathrm{f}})^{l}$
w_N	P_l	$K_{\mathrm{f}}(A+BK_{\mathrm{f}})^{l-1}$
$\vdots$	$\vdots$	$\vdots$
w_{N+l-2}	P_2	$K_{\mathrm{f}}(A+BK_{\mathrm{f}})$
w_{N+l-1}	P_1	K_{f}

3.B.3 Difference

From the observations given above, the two approaches implement the same control inputs up to $N-1$, as in (3.50) and (3.52). However, they are different after time N. The control input at time $N+l$ depends on the terminal states of the nominal prediction $\hat{x}_N$ and the disturbances $w_0, \ldots, w_{N+l-1}$. Table 3.2 shows the coefficients of these terms in (3.51) and (3.53). It is assumed that $l < N$ here, but the case $l \geq N$ does not change the discussion below.

By comparing the coefficients listed in the second and third columns of Table 3.2, it can be seen that some $u_{N+l|N+l}$ cannot be represented with the form of (3.53) using any choice of K_{f}. Assuming $w_0 = \cdots = w_{N+l-3} = 0$, we only need to look at the bottom two rows of the table. The last row requires $K_{\mathrm{f}} = P_1$, but we can easily choose P_2 such that $P_2 \neq P_1(A+BP_1) = K_{\mathrm{f}}(A+BK_{\mathrm{f}})$. This discussion shows that constraint tightening is not subsumed by FPO.

One exception is the 1-state, 1-input, 1-disturbance system with a nilpotency requirement. In that case, nilpotency requires $P_j = 0$, $(j = 2, \ldots, N-1)$ for the CT column, and $(A+BK_{\mathrm{f}}) = 0$ for the FPO column, so the argument above does not

hold, but this exception is a very restrictive case and is of little importance.

Chapter 4

Robust Receding Horizon Control for Trajectory Planning

This chapter presents a receding horizon controller (RHC) that can be used to design trajectories for a UAV operating in an environment with disturbances. The algorithm is a combination of the constraint tightening presented in Chapter 3 and the receding horizon multi-resolution planning presented in Chapter 2. In particular, it uses constraint tightening to achieve robustness to external disturbances, an invariant set to ensure safety in the presence of changes to the environment, and a cost-to-go function to generate an intelligent trajectory around known obstacles. The approach chooses online a robust control invariant admissible set as a terminal set that does not need to be a target set of the overall guidance problem. This result extends previous work in two ways: the vehicle is guaranteed to remain safe under the influence of disturbances; and much longer robust trajectories can be constructed online. The full algorithm is demonstrated in several numerical simulations and hardware experiments.

4.1 Introduction

The focus of this chapter is a class of problems, called "target reach," which are of particular interest for the guidance of unmanned aerial vehicles and the trajectory/activity planning for autonomous rovers. In contrast to regulation problems,

a terminal target set must be reached by the trajectory while meeting various constraints on the way. The algorithm presented in [52] assumed that the target set is reachable over the planning horizon. However, the computation required to design a long trajectory could be excessive if this requires to solve large optimization problems online. One approach is to terminate the optimization when a time limit is reached. When the algorithm has a feasible candidate solution, the optimization could be stopped at any time. However, this will degrade the performance of the closed-loop system, and it is still necessary to find an initial feasible solution. If the controller is required to design a long trajectory from the initial states that are not known *a priori,* then finding this initial feasible solution online could be difficult.

This chapter extends the constraint tightening approach to address these computational issues. In particular, the new algorithm presented in this chapter does not explicitly require that the system states be able to reach the target set over the planning horizon. Instead, the controller only requires that the states can be driven to a robust control invariant set, which can be updated as the system evolves. This approach also represents an extension of the concept of a *basis state* in which the system can safely remain for an indefinite period of time [15, 62, 70]. Note that the robust control invariant set used in this chapter needs not be at or around the target, as is common in other MPC methods [45, 47, 102], and this enables the use of very short planning horizons. The approach is combined with a *cost-to-go* function, which can provide a good estimate of the path beyond the planning horizon to the goal [32]. As a result, the algorithm can be used to solve much longer robust paths than the approach in [52].

The chapter is organized as follows. Section 4.2 extends the algorithm presented in Chapter 3 so that the target set is not necessarily reached. Section 4.3 presents several simulation results, followed by hardware experimental results in Section 4.4.

4.2 Robust Constrained RHC Algorithm

The problem of interest has the overall goal of *reaching the target set* while robustly *maintaining feasibility.* However, in order to maintain the feasibility, reaching the overall goal could be aborted. The algorithm relaxes the constraints that the target set must be reached in the planning horizon, allowing the controller to use a short planning horizon. The computational burden then becomes less severe even when the target set is far. A cost-to-go function provides a good estimate of the remainder of the path to reach the target, even in a complicated environment.

Various constraints are imposed in the problem, such as turning rate limits and bounds on the vehicle speed, and target regions and no-fly zones are included in the environment. The proposed algorithm modifies these constraints to ensure that the online optimization remains feasible even when the vehicle is acted upon by unknown, but bounded, disturbances. In order to maintain safety [70] of the vehicle under the changes in the environment, additional constraints are added that require that *some* robust control invariant admissible set $\mathcal{R}$ is reachable over the short planning horizon. This ensures that the feasibility of the optimization at time k implies the feasibility at the next time $k+1$, resulting in the robust feasibility.

4.2.1 Problem Statement

The LTI system dynamics are the same as in Chapter 3

$$\boldsymbol{x}_{k+1} = A\boldsymbol{x}_k + B\boldsymbol{u}_k + \boldsymbol{w}_k \tag{4.1}$$

$$\boldsymbol{y}_k = C\boldsymbol{x}_k + D\boldsymbol{u}_k \in \mathcal{Y} \tag{4.2}$$

$$\boldsymbol{w}_k \in \mathcal{W} \tag{4.3}$$

where $\boldsymbol{x}_k$ is the state vector, $\boldsymbol{u}_k$ is the input vector, and $\boldsymbol{w}_k$ is the disturbance vector. The disturbance $\boldsymbol{w}_k$ is random but lies in the bounded set $\mathcal{W}$, which is assumed to be known. The constraint (4.2) captures the constraints on the states, the inputs, or combinations of both by using the general output vector $\boldsymbol{y}_k$.

The general objective function takes the form

$$J = \sum_{k=0}^{\infty} l(\boldsymbol{u}_k, \boldsymbol{x}_k, \mathcal{X}_{\mathrm{F}}) \tag{4.4}$$

where $l(\cdot)$ is a stage cost function and $\mathcal{X}_{\mathrm{F}}$ is a target set into which the state $\boldsymbol{x}$ is to be driven. In the receding horizon framework, the optimization is performed over a finite horizon; then, the first control input is executed; the new state is measured and the new optimization is repeated until the system reaches the target set.

4.2.2 Algorithm

MPC solves the optimization problem online as the new information on the states becomes available. The control inputs are developed for a finite horizon of N steps, where the prediction of a value at time $(k+j)$ made at time k is denoted by subscript $(\cdot)_{k+j|k}$. The online optimization problem at time k is defined as:

$$J^* = \min_{\boldsymbol{u}_{\cdot|k}, \mathcal{S}_k} \left\{ \sum_{j=0}^{N-1} l\big(\boldsymbol{u}_{k+j|k}, \boldsymbol{x}_{k+j|k}, \mathcal{X}_{\mathrm{F}}\big) + f\big(\boldsymbol{x}_{k+N|k}, \mathcal{X}_{\mathrm{F}}\big) \right\} \tag{4.5}$$

subject to

$$\boldsymbol{x}_{k+j+1|k} = A\boldsymbol{x}_{k+j|k} + B\boldsymbol{u}_{k+j|k}, \quad \forall j^- \tag{4.6}$$

$$\boldsymbol{y}_{k+j|k} = C\boldsymbol{x}_{k+j|k} + D\boldsymbol{u}_{k+j|k} \ \in \mathcal{Y}_j, \quad \forall j^- \tag{4.7}$$

$$\boldsymbol{x}_{k|k} = \boldsymbol{x}_k \tag{4.8}$$

$$\boldsymbol{x}_{k+N|k} \in \mathcal{S}_k \tag{4.9}$$

$$\mathcal{S}_k = \mathcal{R}_k \sim L_{N-1}\mathcal{W} \tag{4.10}$$

$$\forall \boldsymbol{x} \in \mathcal{R}_k \Rightarrow \begin{cases} A\boldsymbol{x} + B\kappa(\boldsymbol{x}) + L_{N-1}\boldsymbol{w} \in \mathcal{R}_k, & \forall \boldsymbol{w} \in \mathcal{W} \\ C\boldsymbol{x} + D\kappa(\boldsymbol{x}) \in \mathcal{Y}_{N-1}. \end{cases} \tag{4.11}$$

The matrices $L_0, \ldots, L_{N-1}$ and the output constraint sets $\mathcal{Y}_0, \ldots, \mathcal{Y}_{N-1}$ are defined in (3.10)–(3.13). As noted in Section 3.2, $\forall j$ implies $\forall j = 0, \ldots, N-1$, and $\forall j^-$ implies

$\forall j = 0, \ldots, N-2$.

The constraint (4.9) of this algorithm ensures that the terminal step of the plan enters the invariant set characterized by (4.11). The set $\mathcal{S}_k$ in (4.9) is called a *safety* set, because once the vehicle enters $\mathcal{S}_k$, it can remain in the set $\mathcal{R}_k$ indefinitely without violating any constraints. The key difference from the formulation in Chapter 3 is that the terminal set $\mathcal{S}_k$ is a decision variable that is calculated online as the system evolves and new information becomes available. This is important when the vehicle is operated in a cluttered environment with limited initial information and the offline calculation of a general invariant set is intractable.

Theoretically, using the maximal robust control invariant admissible set $\mathcal{C}_\infty$ as a terminal set, where $\mathcal{C}_\infty$ is the greatest feasible invariant set under any nonlinear feedback $\kappa(\boldsymbol{x})$, provides a larger set of initial states from which the optimization has a feasible solution. However, the calculation of the maximal robust control invariant set could be very computationally intensive, even for offline computation, unless the problem setup is very simple (*e.g.,* double integrator with a few constraints). In the trajectory optimization problems, it is usually not feasible to precalculate the exact maximal robust control invariant admissible set and use it in the online MPC optimization. In such cases, finding a good robust control invariant set online is crucial to maintaining feasibility and achieving a good performance. The proposed approach parameterizes the invariant set and solves for a simple robust control invariant admissible set $\mathcal{R}_k$ in the optimization at time k. The invariant set could be parameterized in several ways, depending on the dynamics of the vehicle, and Section 4.3 addresses this issue in more detail.

The algorithm uses the target set $\mathcal{X}_\mathrm{F}$ in the objective function, and there is no requirement that the target set is reached. The function $f(\cdot)$ is the terminal penalty that represents the distance from the safety set $\mathcal{S}_k$ to the goal $\mathcal{X}_\mathrm{F}$. Hence, minimization will drive the system to the goal if possible. This function $f(\cdot)$ is called a cost-to-go function in the receding horizon framework, and one simple example is a two-norm distance between the safety set and the goal. However, choosing a good cost-to-go function could significantly enhance the performance of the planning

Algorithm 4.1 RSBK algorithm

1: Given a disturbance feedback controller and a set of cost points $\boldsymbol{r}_{\text{cp}}$, calculate the output constraint sets $\mathcal{Y}_j$ and a cost map $c(\boldsymbol{r}_{\text{cp}})$ that can be used to evaluate the cost-to-go function $f(\cdot)$
2: **for** $k = 0$ to $k = \infty$ **do**
3: Take a measurement of the current states $\boldsymbol{x}_k$
4: **if** the knowledge of the environment has changed **then**
5: Redo line 1
6: **end if**
7: Formulate a MILP problem using the stored values from line 1
8: Solve optimization (4.5)–(4.11) and obtain the control inputs $\boldsymbol{u}_{k+j|k}$
9: Apply the first step of the optimal control sequence to the system (4.1): $\boldsymbol{u}_k = \boldsymbol{u}_{k|k}$
10: Go to the next time step
11: **end for**

system [32], especially when the operating environment is complicated. Simulation results in Section 4.3 highlight the significance of this cost-to-go function.

The complete algorithm is summarized in Algorithm 4.1. The algorithm is called robust safe but knowledgeable (RSBK) algorithm in this thesis.

4.2.3 Algorithm Properties

Theorem 4.1 (Robust Feasibility). *The system* (4.1) *controlled by Algorithm 4.1 satisfies the output constraint* (4.2) *under the action of the bounded disturbance* (4.3) *for all positive k, if the optimization* (4.5)–(4.9) *at initial step $k = 0$ is feasible.*

Proof. The proof is based on a recursion and is similar to the proof of Theorem 3.1. The only difference is that the sets $\mathcal{R}_k$ and $\mathcal{S}_k$ are decision variables, and must be able to be constructed from the solution of the previous time step. A candidate solution is given by

$$\hat{\boldsymbol{u}}_{k+j+1|k+1} = \boldsymbol{u}_{k+j+1|k} + P_{j+1}\boldsymbol{w}_k, \quad \forall j^- \tag{4.12a}$$

$$\hat{\boldsymbol{x}}_{k+j+1|k+1} = \boldsymbol{x}_{k+j+1|k} + L_j\boldsymbol{w}_k, \quad \forall j \tag{4.12b}$$

$$\hat{\boldsymbol{u}}_{k+N|k+1} = \kappa\big(\hat{\boldsymbol{x}}_{k+N|k+1}\big) \tag{4.12c}$$

$$\hat{\boldsymbol{x}}_{k+N+1|k+1} = A\hat{\boldsymbol{x}}_{k+N|k+1} + B\hat{\boldsymbol{u}}_{k+N|k+1} \tag{4.12d}$$

$$\hat{\mathcal{R}}_{k+1} = \mathcal{R}_k \tag{4.12e}$$

$$\hat{\mathcal{S}}_{k+1} = \mathcal{S}_k \tag{4.12f}$$

Since $\mathcal{R}_k$ is a robust invariant set, $\hat{\mathcal{R}}_{k+1}$ is also a robust invariant set that satisfies (4.11). Thus, the candidate set $\hat{\mathcal{S}}_{k+1} = \mathcal{S}_k$ is also feasible. Therefore, $\boldsymbol{x}_{k+N|k} \in \mathcal{S}_k$ at time k implies $\hat{\boldsymbol{x}}_{k+N+1|k+1} \in \mathcal{S}_k$ at time $k+1$, satisfying (4.9). The rest of the proof is identical to the proof of Theorem 3.1. □

Remark 4.1. The algorithm does not require that the target region $\mathcal{X}_\mathrm{F}$ is reachable over the planning horizon N. The horizon N could be very short, resulting in a computationally fast algorithm.

Remark 4.2. In order to recursively prove the robust feasibility, the algorithm requires the existence of the initial feasible solution. However, because the horizon length N is much shorter than in previous algorithms, this approach can find an initial feasible solution much faster than the full horizon approach.

Remark 4.3. If the candidate control K_j is nilpotent so that $L_{N-1} = 0$, then the set $\mathcal{R}_k = \mathcal{S}_k$ is a *nominal control invariant set*

$$\forall \boldsymbol{x} \in \mathcal{R} \;\Rightarrow\; \begin{cases} A\boldsymbol{x} + B\kappa(\boldsymbol{x}) \in \mathcal{R} \\ C\boldsymbol{x} + D\kappa(\boldsymbol{x}) \in \mathcal{Y}_{N-1} \end{cases}$$

which is much easier to compute. One simple nominal invariant set for fixed-wing aircraft is a loiter circle, or for rotorcraft, any point with zero velocity is invariant [70].

Remark 4.4. In contrast to the nominal safety approach [70] that assumes no disturbance (i.e., $\mathcal{W}^p = 0$), the algorithm presented here never fails to find a feasible solution under the action of bounded disturbances. Furthermore, the number of control variables is the same as the nominal algorithm. By over-bounding the Pontryagin difference operation in (3.13) and (4.10), the algorithm will have the same number of constraints [102].

Remark 4.5. The RSBK algorithm is an *anytime algorithm*, that is, the optimization

Algorithm 4.2 Modified RSBK algorithm

1: Given a disturbance feedback controller, calculate the output constraint sets $\mathcal{Y}_j$ and a cost map $\boldsymbol{r}_{\text{cp}}$, $c(\boldsymbol{r}_{\text{cp}})$ that can be used to evaluate the cost-to-go function $f(\cdot)$
2: **for** $k = 0$ to $k = \infty$ **do**
3: Take a measurement of the current states $\boldsymbol{x}_k$
4: **if** the knowledge of the environment has changed **then**
5: Redo line 1
6: **end if**
7: Formulate a MILP problem using the stored values from line 1
8: Solve optimization (4.13), (4.6)–(4.11), (4.14) and obtain the control inputs $\boldsymbol{u}_{k+j|k}$
9: Apply the first step of the optimal control sequence to the system (4.1): $\boldsymbol{u}_k = \boldsymbol{u}_{k|k}$
10: Go to the next time step
11: **end for**

can be stopped at anytime. In such a case, however, a feasible solution is always available. This follows because a candidate feasible solution can be always constructed from the previous feasible (not necessarily optimal) solution. As shown in (4.12), the calculation of a candidate solution is simple and involves 1) shifting the previous plan by one time step, 2) adding a disturbance feedback sequence, and 3) appending a terminal control input using κ at the terminal step of the plan.

With a slight modification to the RSBK algorithm, it can be shown that the algorithm decreases the cost monotonically. The modified algorithm introduces another variable $\boldsymbol{s}_k$ to evaluate the cost-to-go function, replacing $\boldsymbol{x}_{k+N|k}$ in (4.5). The formulation requires that the staged cost is 0, giving the following optimization

$$J^* = \min_{\boldsymbol{u}(\cdot),\, \boldsymbol{s}_k,\, \mathcal{S}_k} f\big(\boldsymbol{s}_k, \mathcal{X}_{\text{F}}\big) \tag{4.13}$$

subject to

$$\text{(4.6)–(4.11)}$$

$$\boldsymbol{s}_k \in \mathcal{S}_k. \tag{4.14}$$

This variable $\boldsymbol{s}_k$ is required to be in the safety set $\mathcal{S}_k$ and measures how good the

terminal safety set is.

Theorem 4.2 (Monotonically Decreasing Cost). *The objective function* (4.13) *monotonically decreases over time k in Algorithm 4.2.*

Proof. Consider the candidate solution given in (4.12) and

$$\hat{\boldsymbol{s}}_{k+1} = \boldsymbol{s}_k. \tag{4.15}$$

Then,

$$\hat{\boldsymbol{s}}_{k+1} = \boldsymbol{s}_k \in \mathcal{S}_k = \hat{\mathcal{S}}_{k+1}$$

so that the candidate solution (4.12) and (4.15) for time $k+1$ is feasible. Furthermore, (4.15) shows that this candidate solution at time $k+1$ gives the same objective value in (4.13) as the solution obtained in the previous time k. Therefore, by successive optimization, the cost is non-increasing (i.e., monotonically decreasing). □

4.3 Simulation Results

This section presents several simulation results that highlight the extensions in this new RSBK algorithm. The simulation uses rotorcraft, but the algorithm easily extends to other vehicles, such as fixed-wing UAVs. The motion of the vehicle is assumed to be in 2D.

4.3.1 Vehicle Model

The rotorcraft dynamics can be approximated by a double integrator with constraints on speed and acceleration.

$$\begin{bmatrix} \boldsymbol{r}_{k+1} \\ \boldsymbol{v}_{k+1} \end{bmatrix} = A \begin{bmatrix} \boldsymbol{r}_k \\ \boldsymbol{v}_k \end{bmatrix} + B\boldsymbol{a}_k + \boldsymbol{w}_k \tag{4.16}$$

$$\text{with} \qquad A = \begin{bmatrix} I_2 & \Delta t\, I_2 \\ O_2 & I_2 \end{bmatrix}, \qquad \text{and} \qquad B = \begin{bmatrix} \frac{(\Delta t)^2}{2} I_2 \\ \Delta t I_2 \end{bmatrix}$$

where $\boldsymbol{r}$, $\boldsymbol{v}$, and $\boldsymbol{a}$ are the position, velocity, and acceleration vector respectively. I_2 and O_2 express an identity matrix and a zero matrix of size 2 respectively. The disturbance $\boldsymbol{w}_k$ enters as a disturbance on the acceleration, and can be written as

$$\boldsymbol{w}_k \in \mathcal{W} = \left\{ \boldsymbol{w} \mid \boldsymbol{w} = B\boldsymbol{n},\ \boldsymbol{n} \in \mathbb{R}^2,\ \|\boldsymbol{n}\|_\infty \leq w_{\max} \right\}. \tag{4.17}$$

Output constraints

The general output constraints include the obstacle avoidance

$$[I_2,\ O_2]\boldsymbol{x}_{k+j|k} \notin \mathcal{O}$$

where $\mathcal{O} \subset \mathbb{R}^2$ expresses no-fly zones that the vehicle is not allowed to enter, the maximum speed

$$\left\| [O_2,\ I_2]\boldsymbol{x}_{k+j|k} \right\|_2 \leq v_{\max}, \tag{4.18}$$

and the maximum acceleration command

$$\left\| \boldsymbol{a}_{k+j|k} \right\|_2 \leq a_{\max} \tag{4.19}$$

where $v_{\max}$ and $a_{\max}$ are the maximum speed and acceleration for the rotorcraft. In this section, a simple two-step nilpotent controller is used to tighten the constraints. The nilpotent controller K for this system is analytically obtained from $(A+BK)^2 = 0$, which produces

$$K = \left[-\frac{1}{\Delta t^2} I_2,\ -\frac{3}{2\Delta t} I_2 \right].$$

Obtaining P_j's through $P_{j+1} = K(A+BK)^j$ and performing constraint tightening (3.13) give the following constraint set [52, 103]

$$[I_2,\ O_2]\boldsymbol{x}_{k+j|k} \notin \mathcal{O} \oplus \alpha_j \mathcal{B} \tag{4.20}$$

$$\left\|[O_2,\ I_2]\boldsymbol{x}_{k+j|k}\right\|_2 \leq v_{\max} - \beta_j \tag{4.21}$$

$$\left\|\boldsymbol{a}_{(|k}+j\right\|_{k)}2 \leq a_{\max} - \gamma_j \tag{4.22}$$

where the set $\mathcal{B}$ represents a 2D unit box, i.e., $\mathcal{B} = \{x \in \mathbb{R}^2 \mid \|x\|_\infty \leq 1\}$, and

$$\begin{aligned}
&\alpha_0 = 0 \\
&\alpha_1 = \|[1\ 0\ 0\ 0]L_0 B\|_1 w_{\max} \\
&\alpha_j = \alpha_1 + \|[1\ 0\ 0\ 0]L_1 B\|_1 w_{\max}, \quad j \geq 2 \\
&\beta_0 = 0 \\
&\beta_1 = \sqrt{2}\,\|[0\ 0\ 1\ 0]L_0 B\|_1 w_{\max} \\
&\beta_j = \beta_1 + \sqrt{2}\,\|[0\ 0\ 1\ 0]L_1 B\|_1 w_{\max}, \quad j \geq 2 \\
&\gamma_0 = 0 \\
&\gamma_1 = \sqrt{2}\,\|[1\ 0]P_1 B\|_1 w_{\max} \\
&\gamma_j = \gamma_1 + \sqrt{2}\,\|[1\ 0]P_2 B\|_1 w_{\max}, \quad j \geq 2.
\end{aligned} \tag{4.23}$$

Note that (4.20) expands the no-fly zones to guarantee robust feasibility. These non-convex constraints are implemented using MILP. The coefficient $\sqrt{2}$ appears when performing the Pontryagin difference between a two-norm bounded set (4.18) or (4.19) and the infinite-norm bounded disturbance set $\mathcal{W}$ (4.17). This is because $\mathcal{W}$ has the maximum magnitude of the length $\sqrt{2}w_{\max}$ in the diagonal directions.

Terminal constraints

In order to reduce the computation load of the online optimization, it is desirable that a simple set is used as $\mathcal{S}_k$. For a rotorcraft UAV, hovering states [70] can be used to form a robust control invariant admissible set $\mathcal{R}_k$. Note that in this example,

$L_{N-1} = 0$ when $N \geq 2$, due to the nilpotency of the controller K. This allows us to use a nominal control invariant set as $\mathcal{S}_k$ with a tightened constraint set, leading to a very simple parametrization. The invariance (4.11) of the set $\mathcal{S}_k$ is implemented by imposing the following hovering constraints in the optimization.

$$\boldsymbol{x}_{k+N+1|k} = \boldsymbol{x}_{k+N|k} \notin \mathcal{O} \tag{4.24}$$

The tightened constraint set $\mathcal{Y}_{N-1}$ corresponds to $\alpha_{N-1}, \beta_{N-1}, \gamma_{N-1}$ in (4.23).

For fixed wing aircraft with a minimum speed bound, a loitering circle can be used as a terminal set. In such case, one simple parameterized invariant set that can be used to generate the terminal constraints is a loitering circle with the minimum turning radius [70]. Examples using fixed-wing aircraft are given later in Chapter 5.

4.3.2 Results

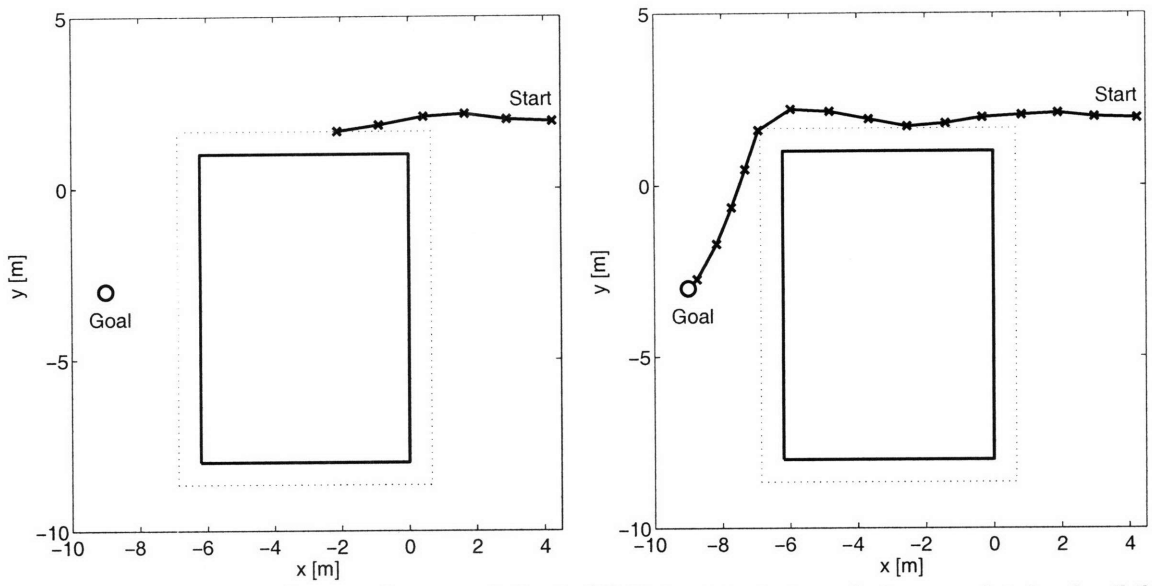
The following values are used in the simulation.

$$\Delta t = 2.6\,\text{seconds}, \qquad v_{\max} = 0.5\,\text{m/s},$$
$$N = 6, \qquad a_{\max} = 0.17\,\text{m/s}^2$$

Comparison with Nominal Safety Approach

The first example compares the two trajectory planning approaches that maintain feasibility of the online optimization using an invariant set. The first approach uses nominal MPC with a nominal invariant set [70]. In this approach, a feasible solution to the optimization problem at time $k+1$ is constructed by combining: 1) the trajectory portion constructed at time k that was not executed; and 2) an extra time step in the nominal invariant set in which the previous trajectory ends. However, the optimization could go infeasible if the vehicle does not reach the state that was predicted in the previous time step, due to disturbance actions.

Figure 4-1(a) shows a trajectory generated with this nominal safety approach. The fifth optimization generates a trajectory that lies near the obstacle boundary. At



(a) Nominal trajectory design – becomes infeasible

(b) Robust trajectory design – maintains feasibility

Figure 4-1: Trajectories generated by the nominal controller and the robust controller. The vehicle starts in the right, and the goal is marked with ○.

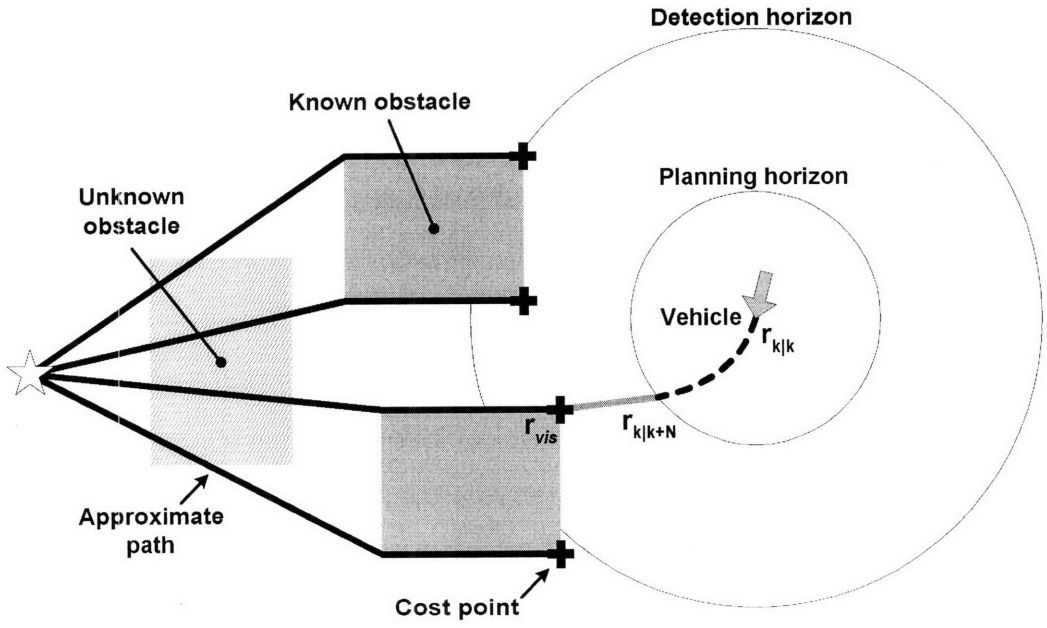


Figure 4-2: Representation of a cost-to-go function [104]

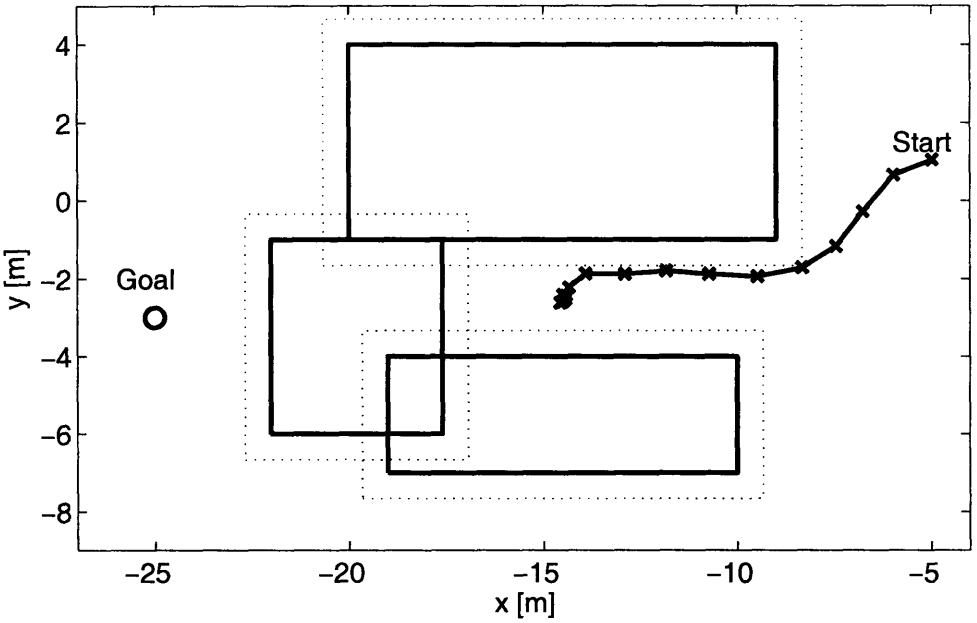
the next time step, the disturbance pushes the vehicle into the infeasible region, and the optimization cannot return a feasible solution.

This issue is successfully resolved by the robust approach presented in this chapter. Figure 4-1(b) shows that the vehicle can reach the goal in spite of the disturbance acting on the system. The constraints were tightened in such a way that no matter where the vehicle reaches after one time step, the feedback correction is possible to maintain feasibility of the online optimizations.

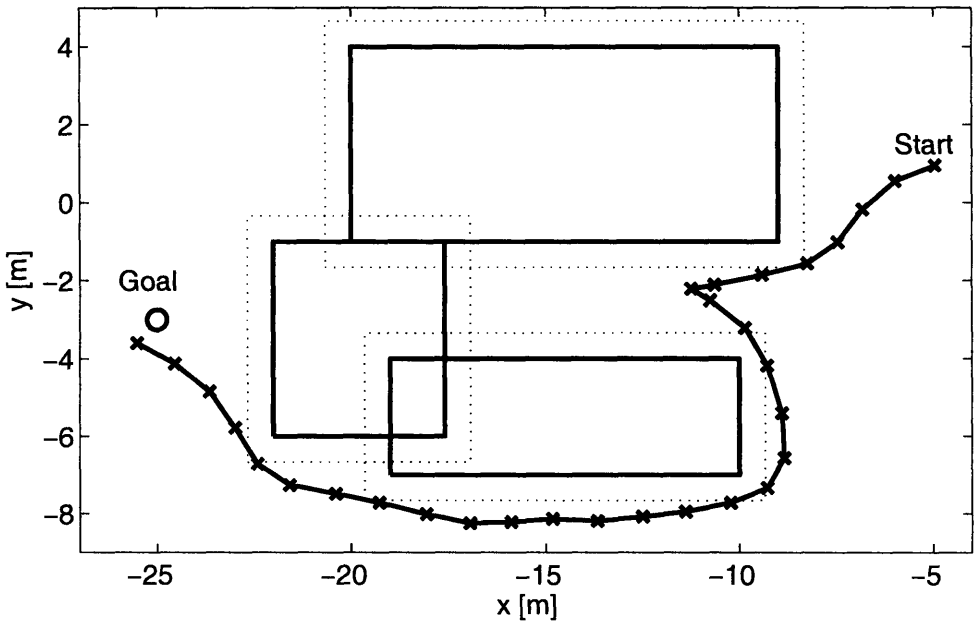
Change in the Environment

The next example demonstrates the performance improvement achieved by the different choice of the terminal penalty function $f(\cdot)$. An intelligent cost-to-go function that represents the future maneuver of the aircraft operated in an obstacle rich field [32, 68, 104] allows RHC to generate a knowledgeable trajectory. As discussed in Chapter 2, for UAV trajectory planning problems with no-fly zones, a cost-to-go evaluated at the terminal state is based on the estimate of the collision free path length from the terminal state to the target region. As shown in Figure 4-2, the cost-to-go estimate is based on the best knowledge of the world, and the new information could become available beyond the detection range.

Figure 4-3 compares trajectories generated with different terminal penalty functions. The obstacle in the left is not initially known. The optimal route is to go through a narrow passage between the other two obstacles. The vehicle finds a new obstacle when it comes within the sensor detection range. The sensor detection range is 8 meters and is larger than the planning horizon ($N = 5$ in this example only). The safety set $\mathcal{S}_k$ is not affected by the new information of the environment and hence is invariant after the obstacle discovery as long as $\mathcal{S}_k$ is within the detection range. Fig. (a) shows that the vehicle remains safe against the pop-up obstacle under the action of the persistent disturbance. In this case, a simple two-norm distance to the goal is used as a cost-to-go function, and the vehicle is trapped in the concave region. In Fig. (b), an intelligent cost-to-go function brings the vehicle out of the entrapment, successfully guiding it to the target.



(a) Simple terminal penalty – feasible but trapped



(b) Intelligent terminal penalty – feasible and not trapped

Figure 4-3: Comparison of two robust safe trajectory planning approaches. Two trajectories are generated with simple terminal penalty and intelligent terminal penalty. The vehicle starts in the right, and the goal is marked with ∘. The disturbance level is 10% of the control magnitude.

Table 4.1: Comparison of the performance for three different disturbance levels.

Disturbance level	Average speed	Steps
0 %	0.50 m/s	26
10 %	0.44 m/s	30
20 %	0.28 m/s	48

4.3.3 Long Trajectory Design

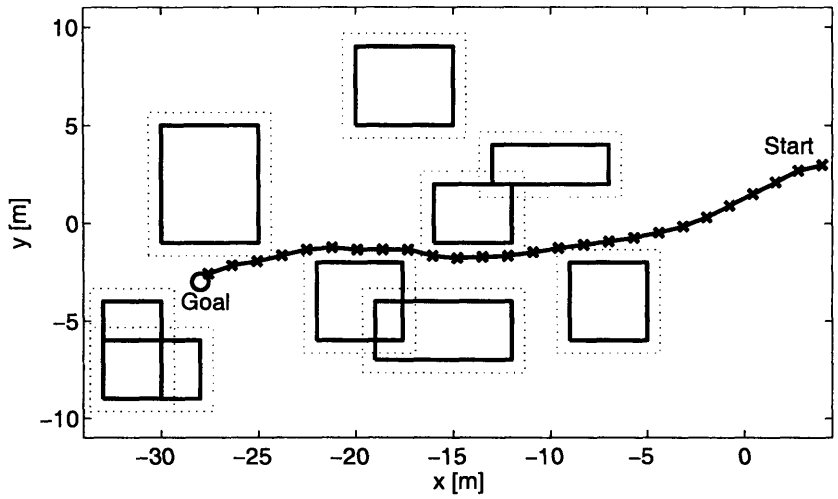
The last simulation results demonstrate that the RSBK algorithm can design a very long trajectory without computational issues. In this example, the target region is far from the current vehicle location. In order for the plan to reach the target set, it is required to make a plan as long as $\sim$30 steps. Designing one long trajectory that reaches this target set is not computationally tractable for real-time applications.

Figure 4-4 shows trajectories generated under three different disturbance levels.

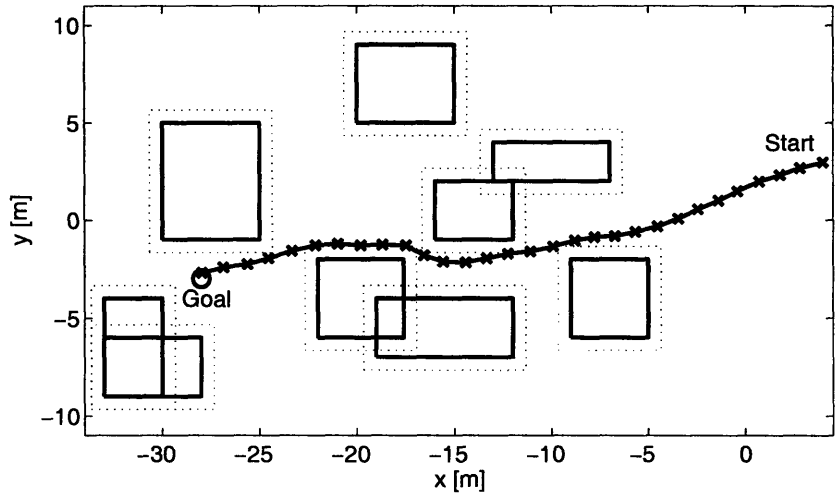
- $w_{\max} = 0$
- $w_{\max} = 0.1\, a_{\max}$
- $w_{\max} = 0.2\, a_{\max}$

In all cases, the RSBK algorithm guided the vehicle to the target, and the average computation time was below 0.2 second. When the disturbance level is 10% of the control authority, the trajectory is similar to the one with no disturbance. However, when the disturbance level is raised to 20% of the control authority, the vehicle takes a different route because the passage in the middle of the figure used by the other plans is too narrow to pass through robustly. A cost-to-go calculation based on the robustified environment $\mathcal{O} \oplus \alpha_{N-1}\mathcal{W}$ does not allow the vehicle to enter the narrow passage where the vehicle could violate the collision avoidance constraints due to a strong disturbance.

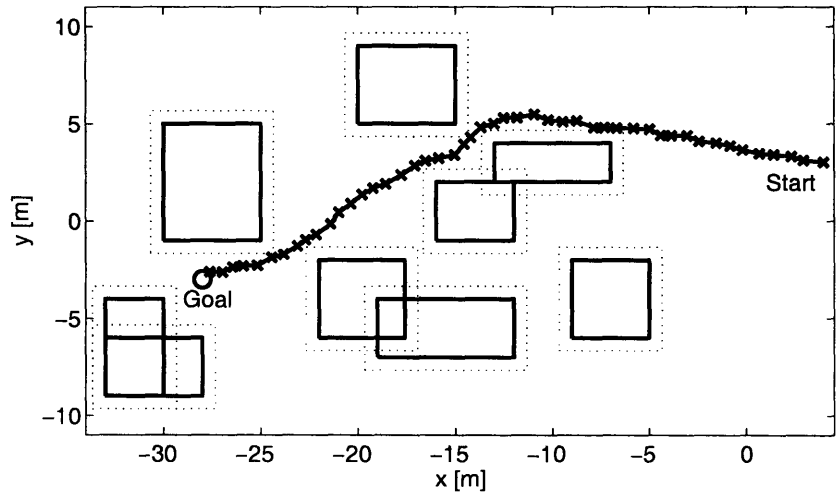
Note that the vehicle moves slowly when the disturbance is strong, as it is expected intuitively. Because more margin must be saved to reject a stronger disturbance, less control authority can be used when generating the trajectory. The hovering state used as a terminal invariant set requires the vehicle be able to stop at the end of each plan



(a) No disturbance.



(b) Disturbance level 10%.



(c) Disturbance level 20%.

Figure 4-4: Long trajectories generated by RSBK algorithm. The vehicle starts at the right, and the goal is marked with ○. The obstacles are expanded in the RSBK calculation to account for the avoidance check imposed only at discrete time steps.

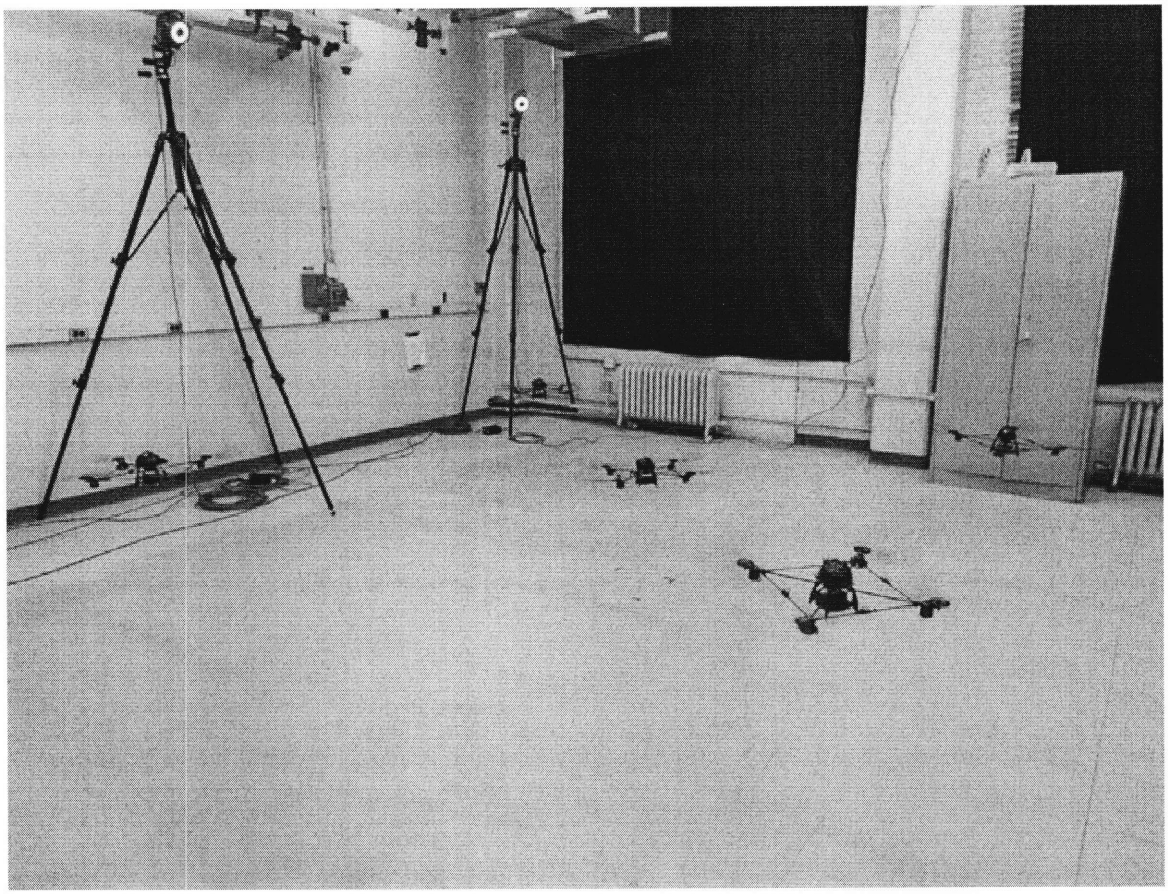


Figure 4-5: Quadrotor testbed using Vicon system [105, 108]

using the small control authority available in the prediction. Table 4.1 summarizes the result. The average speed becomes significantly smaller when the disturbance level is increased from 10% to 20%. The number of steps required to reach the target set is significantly longer with the 20% disturbance level, partly because of the longer route it chooses, but mainly due to the reduced speed.

4.4 Experimental Results

The RSBK algorithm has been tested in the 3D environment using an unique quadrotor testbed developed at the Aerospace Controls Laboratory of MIT.

4.4.1 Quadrotor Testbed

This section briefly describes the quadrotor testbed. More details are available in the recent article [105]. As a sensing device, the testbed uses a Vicon motion capture system [106], which are the cameras shown in background of Figure 4-5. The Vicon system provides a position estimate of sub-millimeter accuracy at 100 Hz, and the filtered time difference gives a velocity estimate of 1 cm/s peak-to-peak accuracy. The vehicles are commercially available Draganflyer V Ti Pro [107], and no significant modifications was required to the hardware to fly them autonomously. Several lightweight reflective markers are attached to each vehicle in a unique configuration, which the Vicon system uses to track the position and orientation of each vehicle in the room. The low-level controller is designed to track waypoints which are then provided in real-time by the planner. The waypoint follower was designed using standard LQR techniques, which calculates the motor commands of each rotor off-board and sends them to the quadrotor using an R/C transmitter [105].

4.4.2 Implementation with Non-zero Computation Time

Figure 4-6 shows how the algorithm is implemented, when the computation time is not negligible. The discrete time step k is defined as the time when the control $\boldsymbol{u}_{k|k}$ is implemented. The latest measurement is taken τ seconds before the discrete time, where τ is an upper bound on the computation time. To form $\boldsymbol{x}_{k|k}$, the measured states are propagated using a model of the low-level controller, assuming that the control input $\boldsymbol{u}_{k-1|k-1}$ is still being executed. The propagated states $\boldsymbol{x}_{k|k}$ are then used as the initial condition of the optimization at time k.

4.4.3 Result

The objective of this experiment is to demonstrate that the RSBK algorithm can generate online a long trajectory using receding horizon techniques. Many disturbances sources exist in the hardware experiments, such as air flow, modeling error of the vehicle, sensing noise, communication delay, and imperfect tracking of the low-level

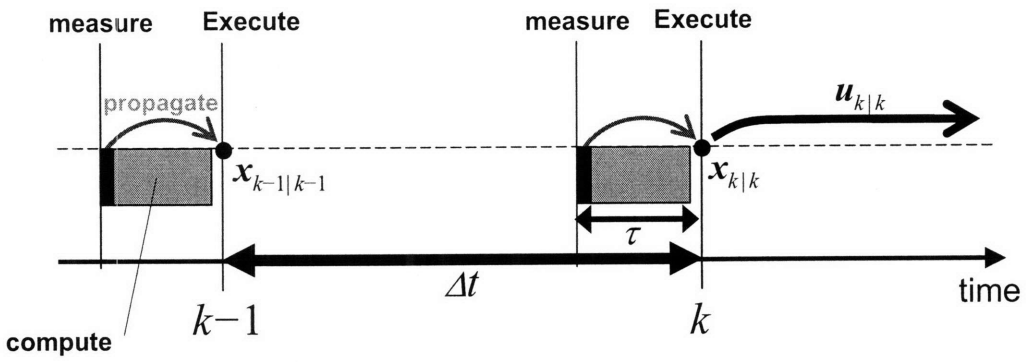


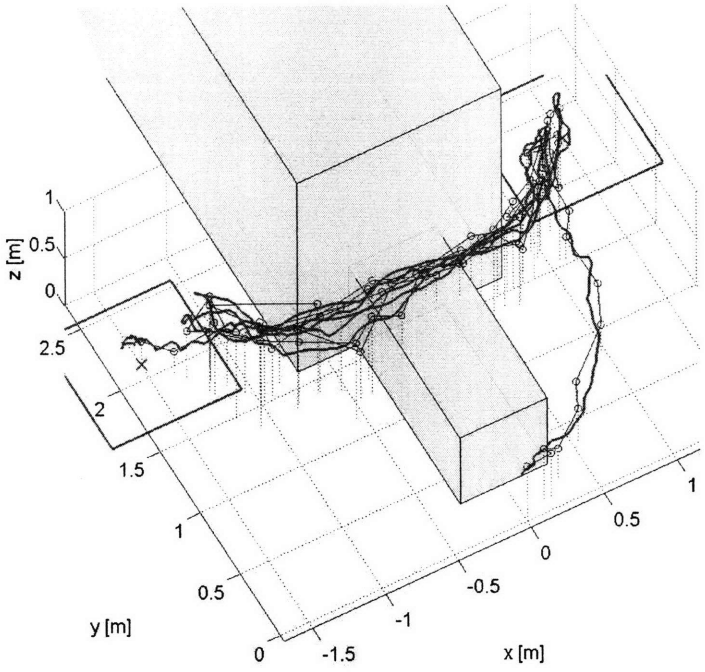
Figure 4-6: Implementation of the algorithm with non-zero computation time

controller. The RSBK algorithm must account for these uncertainties to robustly satisfy the constraints. The following parameters were used in the tests.

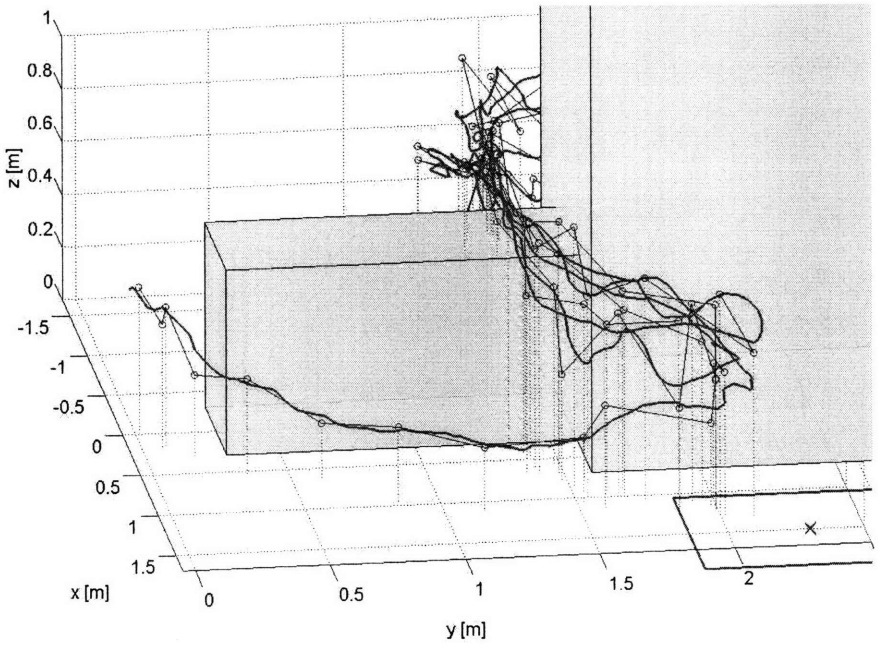
$$\Delta t = 3\,\text{sec}, \qquad \tau = 1.1\,\text{sec}$$
$$v_{\max} = 0.25\,\text{m/s}, \qquad a_{\max} = 0.31\,\text{m/s}^2$$
$$w_r = 0.12\,\text{m}, \qquad w_v = 0.10\,\text{m/s}$$
$$N = 4.$$

In order to demonstrate the robustness of this approach, several long flight tests were conducted. Figures 4-7 to 4-9 show the scenario and the resulting trajectories. The mission is to fly back and forth between two target areas several times. The target areas are squares, 0.9 m on a side, centered at (1.5, 2.3, 0) and (−1.5, 2.1, 0), and are in green lines in the figure (the centers are marked with ×). The vehicle must avoid the obstacles in the middle of the room (a box on the floor next to a large pole). The performance objective penalized the vehicle altitude, and the main objective of minimizing the time of arrival forces the vehicle to fly close to the obstacle boundaries. In scenario #1 and #3, the quadrotor starts around (0, 0, 0.5). The scenario #2 is a continuation of the scenario #1, and the vehicle started around (−1.5, 2.1, 0.2).

The thick red lines are the actual trajectory of the quadrotor recorded at 2 Hz. The planned waypoint commands are marked with ∘ and are connected with blue lines. The low-level controller tracked the planned trajectory reasonably well, and

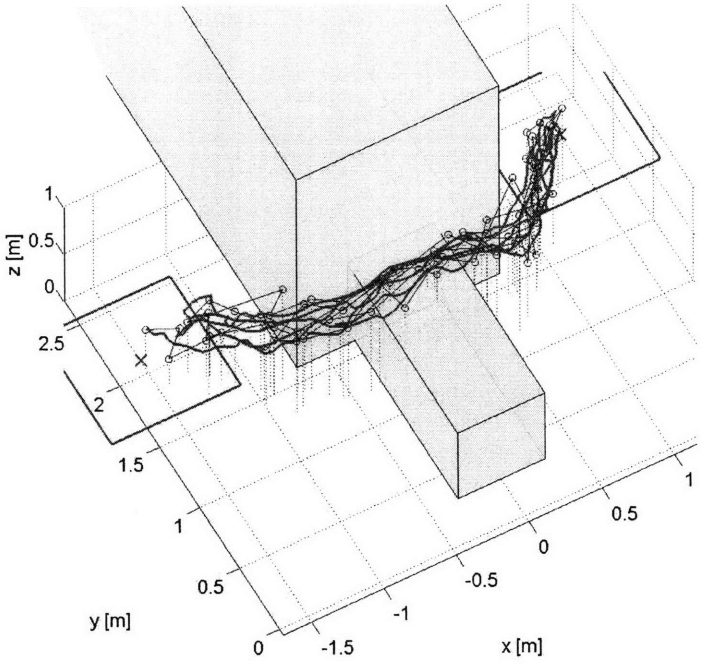


(a) Test run #1. View from above

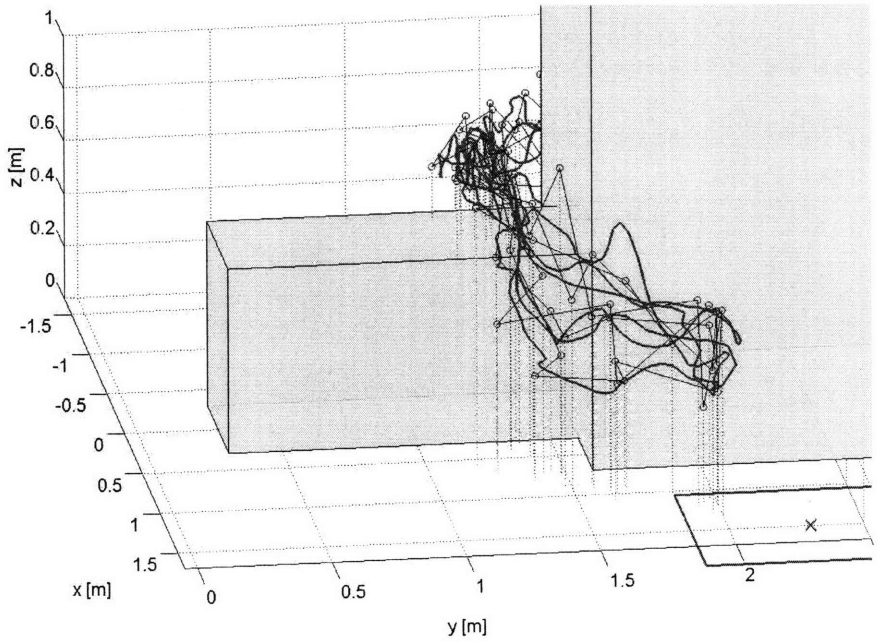


(b) Test run #1. View from $+x$

Figure 4-7: RSBK algorithm on the quadrotor testbed in 3D environment – Test 1

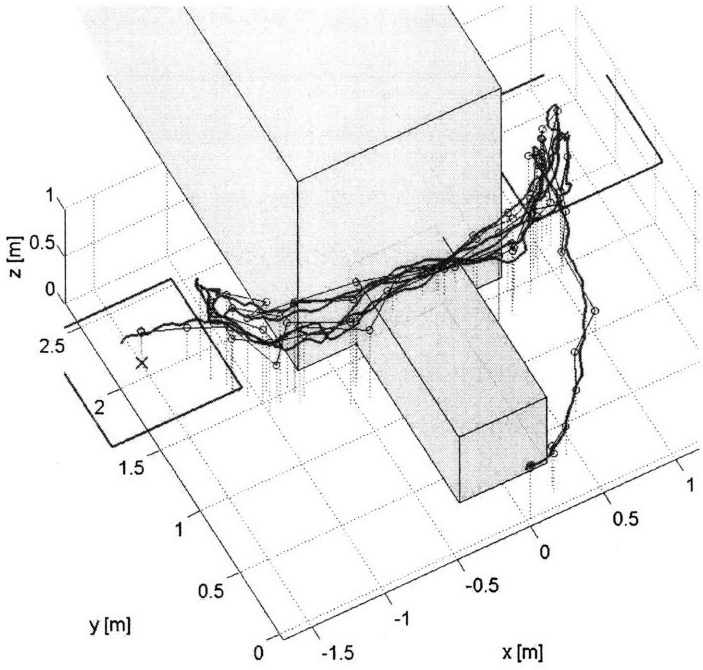


(a) Test run #2. View from above

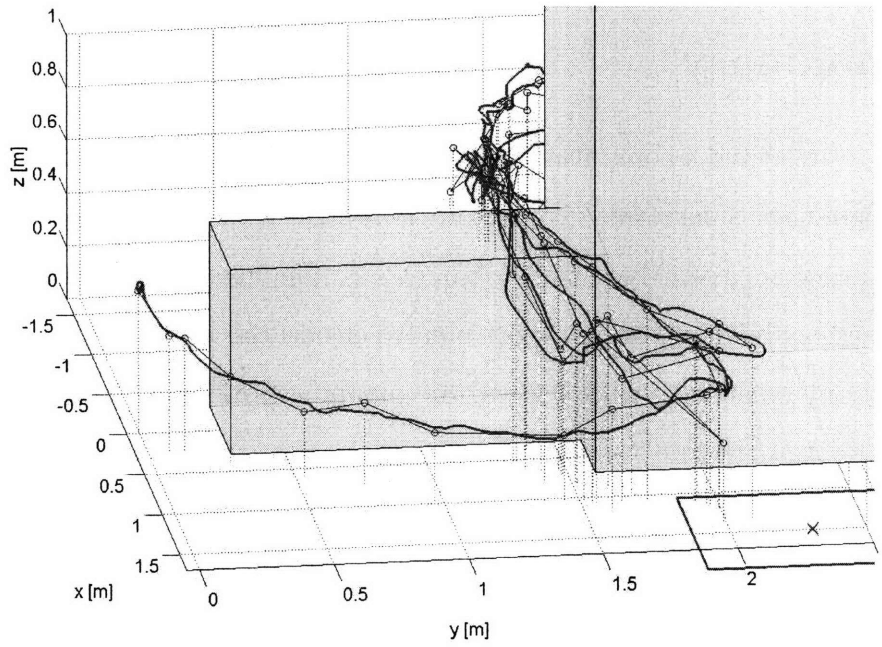


(b) Test run #2. View from $+x$

Figure 4-8: RSBK algorithm on the quadrotor testbed in 3D environment – Test 2



(a) Test run #3. View from above



(b) Test run #3. View from $+x$

Figure 4-9: RSBK algorithm on the quadrotor testbed in 3D environment – Test 3

the difference between the actual and the planned trajectories is captured as a disturbance. In scenario #1 and #3, the vehicle starts by lowering its altitude to minimize the altitude penalty. The planned trajectories between the two targets stay close to the obstacle boundaries, indicating that the trajectories are nearly optimal. Note that the planned trajectory points do not lie on the obstacle boundary because the planner accounts for the disturbances by systematically tightening the constraints (i.e., expanding the obstacles) in the prediction steps. The actual vehicle trajectories show some oscillatory behaviors, which are not modeled in the double integrator model that was used. A better closed-loop model of the low-level controller or a fine tuning of the waypoint follower could reduce the prediction error and improve the smoothness of the actual trajectories. The average MILP computation time was about 1 second using a 2.4GHz laptop with 1GB of RAM. The overall results demonstrate that the RSBK planner can generate three dimensional trajectories online while accounting for disturbances and robustly satisfying constraints.

4.5 Summary

This chapter presented a computationally efficient robust constrained trajectory optimization algorithm. The result extends previous algorithms to allow for much shorter plans that do not necessarily reach the target set. The invariance constraints are imposed at the terminal step of this short plan to ensure the safety of the vehicle under the changes in the environment beyond the planning horizon. A sophisticated cost-to-go function was shown to significantly improve the performance while maintaining feasibility of the optimizations. Simulation and hardware results for a rotorcraft showed that the proposed algorithm safely navigates the vehicle to the target under the action of an unknown but bounded disturbance.

Chapter 5

Decentralized Robust Receding Horizon Control for Multi-vehicle Guidance

This chapter presents a new distributed robust Model Predictive Control algorithm for multi-vehicle trajectory optimization and demonstrates the approach with numerical simulations and multi-vehicle experiments. The technique builds on the robust-safe-but-knowledgeable (RSBK) algorithm in Chapter 4, which is then extended in this chapter for the multi-vehicle case. The key advantage of this RSBK algorithm is that it enables the use of much shorter planning horizons while still preserving the robust feasibility guarantees of previously proposed approaches. The second contribution of this chapter is a distributed version of the RSBK algorithm, which is more suitable for real-time execution. In the distributed RSBK (DRSBK) algorithm, each vehicle only optimizes for its own decisions by solving a subproblem of reduced size, which results in shorter computation times. Furthermore, the algorithm retains the robust feasibility guarantees of the centralized approach while requiring that each agent only have local knowledge of the environment and neighbor vehicles' plans. This new approach also facilitates the use of a significantly more general implementation architecture for the distributed trajectory optimization, which further decreases the delay due to computation time.

5.1 Introduction

When using robust MPC in dynamic environments, fast online computation is needed in response to new information. However, the computation required scales poorly with both the length of the trajectories being planned and the number of vehicles to be planned for. This chapter addresses both of these scalability issues, adopting shorter horizons for scaling with length and distributed computation for scalability with fleet size. The first contribution of this chapter is the extension of *Robust Safe But Knowledgeable* (RSBK) algorithm to the multi-vehicle case. As shown in Chapter 4, this algorithm plans over only a short horizon, terminating in a robust control invariant set that needs not to be near the goal. The main difference is that the constraints and the invariant set in this chapter includes the states/plans of the other vehicles.

For multi-vehicle control, decentralized MPC [57] addresses the computational issue associated with the centralized optimization by breaking the optimization into smaller subproblems, with the rationale that solving many small problems is faster and more scalable than solving one large problem. For multi-vehicle problems, it is natural to divide the problem such that the plan for each vehicle is computed onboard that vehicle, i.e., such that local decisions are made locally. Besides the computational advantages of decentralized MPC, this also offers a reduction in the amount of data that needs to be exchanged between vehicles, and a potentially reduced level of dependency of any individual vehicle.

A second contribution of this chapter is to develop a distributed form of RSBK (DRSBK). The primary computational benefit of the DRSBK algorithm over RSBK is that each vehicle only calculates its own trajectory, which is obtained by solving a subproblem of reduced size. The algorithm creates a queueing order of non-conflicting groups of vehicles [38], where each group optimizes sequentially, while vehicles within a group solve their subproblems in parallel. This does not require iteration, which is crucial for a real-time implementation over a realistic communication network. The chapter also presents a generalization of the implementation architecture for widely

separated teams of vehicles. In particular, we define a local neighborhood of each vehicle to be all other vehicles that could have a direct conflict with that vehicle. By limiting the number of vehicles to consider to only those within a local region of each vehicle, the number of constraints in each subproblem can be significantly reduced. This modification further simplifies the DRSBK computation, but although the plans are only communicated locally, DRSBK is shown to maintain the robust feasibility of the entire fleet. This architecture generalizes the rigid implementation approaches of Refs. [62, 102] to enable some of the vehicles to compute their plans simultaneously, which can significantly reduce the delay incurred.

The chapter is organized as follows. Following the problem setup in Section 5.2, Section 5.3 presents the RSBK algorithm. Section 5.4 extends the RSBK algorithm to the distributed computation using only local information. Section 5.5 shows several simulation results, and Section 5.6 shows experimental results on the hardware testbed.

5.2 Problem Statement

The problem of interest has the overall goal of *reaching the target* while robustly *maintaining feasibility*. In this chapter, p, q, r that are used as an index or superscript denote the vehicle number, subscript k denotes the current time step, and subscript j denotes the prediction step. There are total of n vehicles whose dynamics are decoupled and are described by an LTI model

$$\boldsymbol{x}_{k+1}^p = A^p \boldsymbol{x}_k^p + B^p \boldsymbol{u}_k^p + \boldsymbol{w}_k^p \tag{5.1}$$

for $p = 1, \ldots, n$, where $\boldsymbol{x}_k^p$ is the state vector, $\boldsymbol{u}_k^p$ is the input vector, and $\boldsymbol{w}_k^p$ is the disturbance vector for the p^{th} vehicle. The disturbances $\boldsymbol{w}_k^p$ are unknown but are assumed to lie in known bounded sets

$$\boldsymbol{w}_k^p \in \mathcal{W}^p. \tag{5.2}$$

The environment has obstacles to be avoided and the vehicles have flight envelope limitations. The general output sets $\mathcal{Y}^p$ capture these local constraints of each vehicle $p = 1, \ldots, n$

$$C^p \boldsymbol{x}_k^p + D^p \boldsymbol{u}_k^p \triangleq \boldsymbol{y}_k^p \in \mathcal{Y}^p. \tag{5.3}$$

Vehicles are coupled through the constraints, and a further set of constraints $c = 1, \ldots, n_c$ are applied to the sum of the outputs from each vehicle

$$\begin{aligned} \forall c: \qquad & \boldsymbol{z}_{k,c}^p = E_c^p \boldsymbol{x}_k^p, \quad \forall p = 1, \ldots, n \\ & \sum_{p=1}^{n} \boldsymbol{z}_{k,c}^p \in \mathcal{Z}_c \end{aligned} \tag{5.4}$$

where $\boldsymbol{z}_{k,c}^p$ denotes p's variable that is coupled with other vehicles' variables. For pair-wise collision avoidance constraints, each constraint c has only two nonzero matrices E_c^p and E_c^q, and enforces a minimum separation between that pair of vehicles

$$\|\boldsymbol{r}_k^p - \boldsymbol{r}_k^q\| \geq 2d \tag{5.5}$$

where $\boldsymbol{r}_k^p$ is a position of the vehicle p, and $2d$ is the minimum separation distance. Note that each set $\mathcal{Z}_c$ is non-convex in this case. Finally, the objective of the trajectory optimization is to navigate the vehicles to their assigned targets, and the objective function is the sum of individual costs

$$\boldsymbol{x}_{N^p}^p \in \mathcal{X}_{\mathrm{T}}^p \tag{5.6}$$

$$J = \sum_{p=1}^{n} \sum_{k=0}^{N^p - 1} l^p(\boldsymbol{x}_k^p, \boldsymbol{u}_k^p) \tag{5.7}$$

where N^p is the time of arrival at vehicle p's target $\mathcal{X}_{\mathrm{T}}^p$ and is a variable to be minimized, and l^p is a staged cost of vehicle p.

5.3 Robust Safe but Knowledgeable Algorithm

This section presents a *Robust Safe but Knowledgeable* (RSBK) algorithm presented in Chapter 4 in the multi-vehicle setting. When applied to multi-vehicle control, this algorithm solves a centralized problem, i.e., solving for the plans of all vehicles $p = 1, \ldots, n$ in a single optimization. Section 5.4 discusses how this computation can be separated into a sequence of smaller problems and distributed across the vehicles in the team.

5.3.1 Algorithm Description

Solving a single optimization (5.1)–(5.7) is not tractable when the vehicle flies through complex environment to a distant target, because the complexity of the optimization grows rapidly with the number of steps N^p required to reach the target. Furthermore, the situational awareness can change as the vehicle flies and there could be significant uncertainties in the far future. It is inefficient to devote considerable computational effort to plans for the far future because these are likely to be revised in the light of future learning.

The online MPC optimization develops the control inputs for a short horizon of N steps. To simplify the presentation, let $\boldsymbol{x}_k$ without the vehicle superscript denote the vehicle states of all the vehicles. The optimization $\mathsf{P}(\boldsymbol{x}_k)$ at time k is defined as:

$$J^* = \min_{\substack{\boldsymbol{u}^p_{\cdot|k}, \mathcal{S}^p_k \\ p=1,\ldots,n}} \sum_{p=1}^{n} \Big\{ \sum_{j=0}^{N-1} l^p\big(\boldsymbol{u}^p_{k+j|k}, \boldsymbol{x}^p_{k+j|k}\big) + f^p\big(\boldsymbol{x}^p_{k+N|k}\big) \Big\} \tag{5.8}$$

subject to $\forall p = 1, \ldots, n$, and $\forall j$

$$\boldsymbol{x}^p_{k|k} = \boldsymbol{x}^p_k \tag{5.9}$$

$$\boldsymbol{x}^p_{k+j+1|k} = A^p \boldsymbol{x}^p_{k+j|k} + B^p \boldsymbol{u}^p_{k+j|k} \tag{5.10}$$

$$\boldsymbol{y}^p_{k+j|k} = C^p \boldsymbol{x}^p_{k+j|k} + D^p \boldsymbol{u}^p_{k+j|k} \in \mathcal{Y}^p_j \tag{5.11}$$

$$\boldsymbol{z}^p_{k+j|k,\,c} = E^p_c \boldsymbol{x}^p_{k+j|k} \tag{5.12}$$

$$\sum_{p=1}^{n} \boldsymbol{z}^p_{k+j|k,\,c} \in \mathcal{Z}_{j,\,c}\,,\ \forall c = 1, \ldots, n_c \tag{5.13}$$

$$\boldsymbol{x}^p_{k+N|k} \in \mathcal{S}^p_k \tag{5.14}$$

$$\mathcal{S}^p_k = \mathcal{R}^p_k \sim L^p_{N-1}\mathcal{W}^p \tag{5.15}$$

$$\forall \boldsymbol{x}^p \in \mathcal{R}^p_k \ \Rightarrow \ \begin{cases} A^p\boldsymbol{x}^p + B^p\kappa^p(\boldsymbol{x}^p) + L^p_{N-1}\boldsymbol{w}^p \in \mathcal{R}^p_k,\ \forall \boldsymbol{w}^p \in \mathcal{W}^p \\ C^p\boldsymbol{x}^p + D^p\kappa^p(\boldsymbol{x}^p) \in \mathcal{Y}^p_{N-1} \\ \forall c = 1, \ldots, n_c : \ \sum_{p=1}^{n} E^p_c \boldsymbol{x}^p \in \mathcal{Z}_{N-1,\,c} \\ \forall (\boldsymbol{x}^1, \ldots, \boldsymbol{x}^n) \in \{\mathcal{S}^1_k \times \ldots \times \mathcal{S}^n_k\}. \end{cases} \tag{5.16}$$

The states $\boldsymbol{x}^p_k$ in (5.9) is the measured states of vehicle p. The decision variables are the control inputs $\boldsymbol{u}^p_{\cdot|k}$ and the terminal invariant set $\mathcal{S}^p_k$ that ensures the safety of the vehicle beyond the planning horizon. Note that predictions (5.10) are made using only the nominal system model, with no disturbance. In order to guarantee robustness against disturbances $\boldsymbol{w}$, the sets $\mathcal{Y}^p_j$ are constructed by tightening the original set $\mathcal{Y}^p$ using a linear controller P^p_j that rejects the disturbance

$$\mathcal{Y}^p_0 = \mathcal{Y}^p \tag{5.17a}$$

$$\mathcal{Y}^p_{j+1} = \mathcal{Y}^p_j \sim (C^pL^p_j + D^pP^p_{j+1})\mathcal{W}^p \tag{5.17b}$$

where L^p_j is a state transition matrix

$$L^p_0 = I \tag{5.18a}$$

$$L^p_{j+1} = A^pL^p_j + B^pP^p_{j+1}, \quad \forall j^-. \tag{5.18b}$$

The notation $\forall j^-$ implies $\forall j = 0, \ldots, N-2$, and $\forall j$ implies $\forall j = 0, \ldots, N-1$, as introduced in previous chapters. Equations (5.12) and (5.13) represent the inter-vehicle constraints such as collision avoidance, and more details on the MILP implementation are found in Appendix 5.C. Similar tightening is performed on the coupling constraint

sets in (5.13), allowing uncertainty margin for all subsystems within each constraint ($\forall c = 1, \ldots, n_c$)

$$\mathcal{Z}_{0,c} = \mathcal{Z}_c \tag{5.19a}$$

$$\mathcal{Z}_{j+1,c} = \mathcal{Z}_{j,c} \sim \left(E_c^1 L_j^1 \mathcal{W}_1 \oplus \ldots \oplus E_c^n L_j^n \mathcal{W}_n \right) \tag{5.19b}$$

where the operator $\oplus$ denotes the Minkowski sum [109]. Unlike other robust MPC approaches, the constraint tightening approach does not increase the complexity of the problem and is well-suited for real-time applications. Another advantage of this approach is that the optimization considers the entire range of vehicle dynamics allowed by the constraints (5.11)–(5.13).

The set $\mathcal{S}_k^p$ in (5.14) is called a *safety* set, defined by (5.15). The set $\mathcal{R}_k^p$ is a robust control invariant admissible set [98] that has a property (5.16). The property states that once the vehicle enters the set $\mathcal{R}_k^p$, the vehicle can remain safe indefinitely, satisfying all the constraints using a pre-determined terminal control law $\kappa^p(\boldsymbol{x}^p)$. The vehicle is safe also against any changes in the environment that occur outside of this safety set. This terminal set $\mathcal{S}_k^p$ moves with the vehicle towards the target and therefore a decision variable in the online optimization, as indicated in (5.8). The RSBK algorithm parameterizes the invariant set, and by using nilpotent candidate controllers, which gives $L_{N-1}^p = 0$, it can solve for a simple nominal control invariant admissible set [52]. One simple invariant set for fixed-wing aircraft is a loiter circle, or for rotorcraft, any point with zero velocity is invariant [70]. Detailed examples are given later in Section 5.5. Note that vehicle q's safety set $\mathcal{S}_k^q$ can overlap with vehicle p's path to its safety set $\mathcal{S}_k^p$ without any issues. This is because by the time q reaches $\mathcal{S}_k^q$, vehicle p has already executed the portion that overlaps with $\mathcal{S}_k^q$.

The function $f^p(\boldsymbol{x}_{k+N|k}^p)$ in (5.8) represents the cost-to-go beyond the planning horizon and is associated with the terminal states of the planned trajectory. The cost-to-go function for minimum time problem is

$$f^p\left(\boldsymbol{x}_{k+N|k}^p\right) = \left\| \boldsymbol{r}_{k+N|k}^p - \boldsymbol{r}_{\text{vis}}^p \right\|_2 + c^p(\boldsymbol{r}_{\text{vis}}^p). \tag{5.20}$$

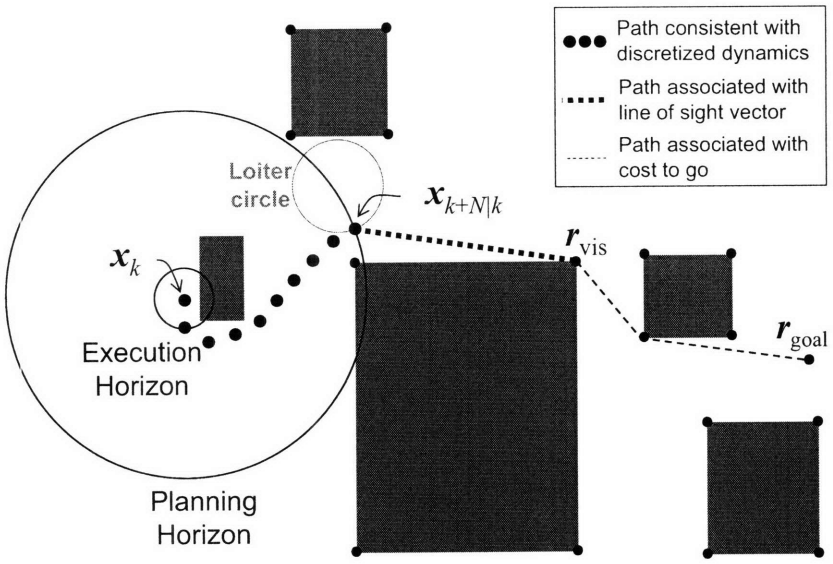


Figure 5-1: Representation of the cost-to-go function showing the three levels of resolution used to approximate a complete path to the goal.

The results presented in this chapter uses a 2D version of the cost-to-go calculation, as shown in Figure 5-1, but the result easily extends to 3D environment considered in Chapter 2.

Given these main components, the overall RSBK algorithm is summarized in Algorithm 5.1. From the optimal solution at time k, the first control input $\boldsymbol{u}^p_{k|k}$ for each vehicle is applied to the system (5.1). At the next time $k+1$, the states of each vehicle $\boldsymbol{x}^p_{k+1}$ are measured, and the optimization is repeated.

5.3.2 Properties

Theorem 5.1 (Robust Feasibility). *The system* (5.1) *controlled by Algorithm 5.1 satisfies all the local and coupling constraints* (5.3)–(5.4) *under the action of bounded disturbances* (5.2) *for all positive* k*, if the optimization* (5.8)–(5.16) *at initial step* $k = 0$ *is feasible.*

Proof. It can be shown that feasibility at time k ensures that a particular candidate

Algorithm 5.1 RSBK algorithm for multiple vehicles

1: Given a disturbance feedback controller, calculate the output constraint sets $\mathcal{Y}_j^p$ through (5.17)–(5.18), the coupling constraint sets $\mathcal{Z}_j$ through (5.19), and a cost map $\boldsymbol{r}_{\text{corner}}$, $c^p(\boldsymbol{r}_{\text{corner}})$ that can be used to evaluate the cost-to-go function $f^p(\cdot)$
2: **for** $k = 0$ to $k = \infty$ **do**
3: Take a measurement of the current states $\boldsymbol{x}_k^p$
4: **if** the knowledge of the environment has changed **then**
5: Redo line 1
6: **end if**
7: Formulate a MILP problem using the stored values from line 1
8: Solve optimization (5.8)–(5.16), (5.20) and obtain the control inputs $\boldsymbol{u}_{k+j|k}^p$
9: Apply control $\boldsymbol{u}_k^p = \boldsymbol{u}_{k|k}^p$ from the optimal sequence to the system (5.1)
10: Go to the next time step
11: **end for**

solution

$$\hat{\boldsymbol{u}}_{k+j+1|k+1}^p = \boldsymbol{u}_{k+j+1|k}^p + P_{j+1}^p \boldsymbol{w}_k^p, \quad \forall j^- \tag{5.21a}$$

$$\hat{\boldsymbol{x}}_{k+j+1|k+1}^p = \boldsymbol{x}_{k+j+1|k}^p + L_j^p \boldsymbol{w}_k^p, \quad \forall j \tag{5.21b}$$

$$\hat{\boldsymbol{u}}_{k+N|k+1}^p = \kappa^p(\hat{\boldsymbol{x}}_{k+N|k+1}^p) \tag{5.21c}$$

$$\hat{\boldsymbol{x}}_{k+N+1|k+1}^p = A^p \hat{\boldsymbol{x}}_{k+N|k+1}^p + B^p \hat{\boldsymbol{u}}_{k+N|k+1}^p \tag{5.21d}$$

$$\mathcal{R}_{k+1}^p = \mathcal{R}_k^p \tag{5.21e}$$

$$\mathcal{S}_{k+1}^p = \mathcal{S}_k^p \tag{5.21f}$$

is feasible at time $k+1$, and hence the optimization at time $k+1$ must be feasible. See Appendix 5.A for more detail. □

Remark 5.1. In order to recursively prove robust feasibility, the algorithm requires the existence of an initial feasible solution. Because the algorithm uses a short planning horizon and does not require the vehicles reach the goal in the first plan, it is typically very easy to find an initial feasible solution, as will be shown in the experimental results Section 5.6. One such initialization is a simple loiter pattern, assuming the vehicles are far enough apart compared to the diameter of the loiter circle. This initialization is much simpler than that required in the previous robust multi-vehicle MPC algorithms [52]. This feature will also be exploited in the distributed form of

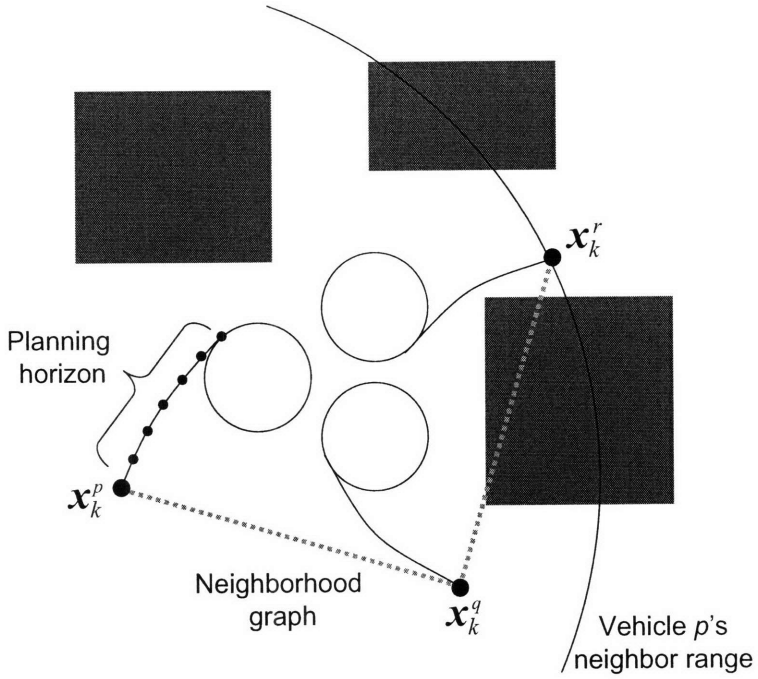


Figure 5-2: The neighborhood of each vehicle is shown by the dashed lines. Each plan terminates in a safety circle.

the algorithm, where initialization can be a significant challenge.

5.4 Distributed RSBK Algorithm

This section presents a distributed version of the RSBK algorithm. In this approach, each vehicle solves a reduced subproblem to determine its control inputs. These optimizations are solved in sequence and the distribution is achieved by having each vehicle exchange its plan information with the other vehicles. A key element of this work is that the vehicles must only exchange information with its neighbors, enabling the local optimization to be based on *local information* [62]. This is important because it reduces the communication requirements and enables the groups to re-plan faster.

5.4.1 Algorithm Description

The basic idea is to include only the vehicles that could have direct conflicts with the vehicle that is planning. Figure 5-2 shows an example with three aircraft. Any plan

of the vehicle r would not have conflict with p's plan because they are far apart. On the other hand, the vehicle q could have a conflict with p if both p and q generate their plans independently and move towards each other. Therefore, p's optimization must include the intention of q, but the vehicle r could be disregarded.

Before presenting the algorithm, several aspects of the notation are defined. First, define vehicle p's neighbor $\mathcal{I}_k^p$ as an ordered set of vehicles whose plans made at time k could have direct conflicts with p's plan made at time k. More formally, $q \in \mathcal{I}_k^p$ if there exist locally feasible solutions $\boldsymbol{u}_{k+j|k}^q, \mathcal{S}_k^q$ and $\boldsymbol{u}_{k+j|k}^p, \mathcal{S}_k^p$ that individually satisfy the local constraints (5.9)–(5.11) and (5.14) for vehicle p and q, but can violate any of the coupling constraints (5.12)–(5.13), (5.15)–(5.16) if combined. For the multi-vehicle collision avoidance problem, a simple implementation of vehicle p's neighborhood is a set of vehicles within distance $2D$ from the vehicle p, where D is the maximum plan length with some margin and is given by

$$D = \sum_{k=0}^{N} (v_{\max} - \beta_k)\Delta t + d + \alpha_{N-1} + 2\rho \tag{5.22}$$

with Δt being the sampling time of the discrete time system, d being the size of the vehicle, ρ being the radius of the loiter circle, α_{N-1} being the margin included for robustness [102], and β_k being the constraint tightening margin for the velocity, whose analytical calculations are given later in (5.50). The arc in Figure 5-2 shows the boundary of p's neighborhood. The communication range of the vehicles is assumed to be larger than $2D$. For each vehicle, all obstacles within range D from the vehicle are assumed to be known. Note that the neighbor set is a function of time k, because the relative position of the vehicles will change over time.

The set $\mathcal{I}_k^p$ also determines the order in which the vehicles calculate their new plans sequentially, although Section 5.4.5 modifies the assumption on this strict ordering. Let $\texttt{pre}(q)$ denote the vehicle ordered prior to the vehicle q, and $\texttt{next}(q)$ denote the vehicle ordered after q. The first and the last element of this set is expressed as $\texttt{first}(\mathcal{I}_k^p)$ and $\texttt{last}(\mathcal{I}_k^p)$, respectively. With the definition of $\mathcal{I}_k^p$, we know a priori that for any two vehicles p and q with $q \notin \mathcal{I}_k^p$ (and hence $p \notin \mathcal{I}_k^q$), the plans of the

two vehicles satisfy the coupling avoidance constraints for all steps over the planning horizon. Hence, even if the subproblem for the vehicle p includes only the plans of its neighbors, all of the coupling avoidance constraints will be satisfied.

The result of this analysis is that only a subset of all n_c coupling constraints need to be considered in each subproblem. Define $\mathcal{C}_k^p \subset \{1, \ldots, n_c\}$ as the set of coupling constraints to be included in subproblem p at time k. Then,

$$\mathcal{C}_k^p = \{c \in 1, \ldots, n_c \ : \exists q \in \mathcal{I}_k^p, [E_c^p \ E_c^q] \neq \mathbf{0}\}. \tag{5.23}$$

This excludes two kinds of constraint irrelevant to p: those that couple p to the vehicles outside its neighborhood, and those that do not involve p at all, i.e., with $E_c^p = \mathbf{0}$.

Let $\mathcal{G}_k$ denote a vehicle graph whose node is a vehicle and edge connects two nodes if the corresponding vehicles are neighbors. If $\mathcal{G}_k$ is a disconnected graph, then $\mathcal{G}_k$ is divided into a set of connected subgraphs. The information of the neighbor sets $\mathcal{I}_k^p$ is shared by the vehicles in the connected graph (or subgraph if $\mathcal{G}_k$ is not connected), so that the vehicles in the connected graph have the consistent information on the planning order. This can be done using only the inter-vehicle communication. Note that the vehicles that belong to different connected graphs do not need to exchange information because there will be no conflict among them.

5.4.2 Algorithm

At time k, the p^{th} vehicle generates its own control inputs $\boldsymbol{u}_{\cdot|k}^p$ by solving the following optimization subproblem $\mathsf{P}^p(\boldsymbol{x}_k^p)$:

$$\min_{\boldsymbol{u}_{\cdot|k}^p, \mathcal{S}_k^p} \sum_{j=0}^{N-1} l^p(\boldsymbol{x}_{k+j|k}^p, \boldsymbol{u}_{k+j|k}^p) + f^p(\boldsymbol{x}_{k+N|k}^p) \tag{5.24}$$

subject to $\forall j$ Eqs. (5.9)–(5.12), (5.14),

$$\boldsymbol{z}_{k+j|k,c}^p + \tilde{\boldsymbol{z}}_{k+j|k,c}^p \in \mathcal{Z}_{j,c}^p, \quad \forall c \in \mathcal{C}_k^p \tag{5.25}$$

$$\forall \boldsymbol{x}^p \in \mathcal{S}_k^p \;\Rightarrow\; \begin{cases} A^p\boldsymbol{x}^p + B^p\kappa^p(\boldsymbol{x}^p) \in \mathcal{S}_k^p \\ C^p\boldsymbol{x}^p + D^p\kappa^p(\boldsymbol{x}^p) \in \mathcal{Y}_{N-1}^p \\ \forall c \in \mathcal{C}_k^p : \sum_{q \in \mathcal{I}_k^p} E_c^q \boldsymbol{x}^q \in \mathcal{Z}_{N-1,\,c} \\ \qquad \forall (\boldsymbol{x}^{p_0}, \ldots, \boldsymbol{x}^{p_n}) \in \{\mathcal{S}_k^{p_0} \times \ldots \times \mathcal{S}_k^{p_n}\} \end{cases} . \tag{5.26}$$

In DRSBK, each vehicle is assumed to use a nilpotent controller P_j^p that makes $L_{N-1}^p = 0$, and therefore (5.15) is not included. The term $\tilde{\boldsymbol{z}}_{k+j|k,\,c}^p$ is a summation of the outputs from the neighbor vehicles and is constant in this local optimization. The term has two components $\forall j$, $\forall c \in \mathcal{C}_k^p$

$$\tilde{\boldsymbol{z}}_{k+j|k,\,c}^p = \sum_{\substack{q \in \mathcal{I}_k^p, \\ \mathtt{ord}(q) < \mathtt{ord}(p)}} \boldsymbol{z}_{k+j|k,\,c}^q + \sum_{\substack{q \in \mathcal{I}_k^p, \\ \mathtt{ord}(q) > \mathtt{ord}(p)}} \boldsymbol{z}_{k+j|k-1,\,c}^q. \tag{5.27}$$

The first term is the summation over the vehicles that have already planned at time k. The second term is for the vehicles that have not planned at time k, so that the prediction made at $(k-1)$ is used. This prediction comes directly from (5.12) in the optimization $\mathsf{P}^q(\boldsymbol{x}_{k-1}^q)$. The original coupling constraint sets $\mathcal{Z}_c$ are modified in the following manner, dividing the tightening process from (5.19b) into intermediate stages for each vehicle

$$\mathcal{Z}_{0,\,c}^{p_n} = \mathcal{Z}_c \tag{5.28a}$$

$$\mathcal{Z}_{j,\,c}^{\mathtt{pre}(q)} = \mathcal{Z}_{j,\,c}^q \sim E_c^q L_j^q \mathcal{W}^q, \qquad \forall j, \quad q \in \mathcal{I}_k^p,\ q \neq p_0 \tag{5.28b}$$

$$\mathcal{Z}_{j+1,\,c}^{p_n} = \mathcal{Z}_{j,\,c}^{p_0} \sim E_c^{p_0} L_j^{p_0} \mathcal{W}_{p_0}, \qquad \forall j^- \tag{5.28c}$$

with $p_0 = \mathtt{first}(\mathcal{I}_k^p)$ and $p_n = \mathtt{last}(\mathcal{I}_k^p)$. (5.28b) tightens the constraints from the vehicle q to $\mathtt{pre}(q)$. This represents that the vehicle $\mathtt{pre}(q)$ saves some margin for the vehicle q so that q can use it to reject the disturbances $\mathcal{W}^q$. (5.28c) tightens the constraints from the prediction step j to $(j+1)$. This represents that the optimization at time k for vehicle p_n saves some margin so that the optimization at time $(k+1)$ for vehicle p_0 can use it to also reject the disturbances.

Algorithm 5.2 DRSBK algorithm

1: Given a disturbance feedback controller, calculate the output constraint sets $\mathcal{Y}_j^p$ through (5.17)–(5.18), the coupling constraint sets $\mathcal{Z}_j^p$ through (5.28), and a cost map $\boldsymbol{r}_{\text{corner}}$, $c^p(\boldsymbol{r}_{\text{corner}})$ that can be used to evaluate the cost-to-go function $f^p(\cdot)$
2: Find a feasible solution of the DRSBK optimization starting from the current states (See Remark 5.2)
3: **for** $k = 1$ to $k = \infty$ **do**
4: **for** each vehicle $p = 1, \ldots, n$ **do**
5: Update the neighbor set $\mathcal{I}_k^p$
6: **end for**
7: **for** each vehicle $p = 1, \ldots, n$, in a predetermined order (*e.g.* $1, \ldots, n$) **do**
8: Gather, by communication, the latest plans $\boldsymbol{z}^q_{\cdot|k,c}$ or $\boldsymbol{z}^q_{\cdot|k-1,c}$ from its neighbors $q \in \mathcal{I}_k^p$
9: Take a measurement of the current states $\boldsymbol{x}_k^p$
10: **if** the knowledge of the environment has changed **then**
11: Redo line 1
12: **end if**
13: Construct a cost map $\boldsymbol{r}_{\text{corner}}$, $c^p(\boldsymbol{r}_{\text{corner}})$
14: Formulate a MILP problem using the stored values from line 1
15: Solve subproblem $\mathsf{P}^p(\boldsymbol{x}_k^p)$ and obtain the control inputs $\boldsymbol{u}^p_{k+j|k}$
16: **end for**
17: Apply control $\boldsymbol{u}_k^p = \boldsymbol{u}^p_{k|k}$ from the optimal sequence to each vehicle p in (5.1)
18: Go to the next time step
19: **end for**

In (5.26), for the vehicles that have already planned at time k, the latest solution $\mathcal{S}_k^q$ is used. For the vehicles that have not planned $\forall q \in \{\texttt{next}(p), \ldots, p_n\}$, the invariant set constructed at the previous step is used, i.e., $\mathcal{S}_k^q = \mathcal{S}_{k-1}^q$.

Algorithm 5.2 shows the full DRSBK algorithm. Note that this algorithm is also a generalization of the two previously published distributed MPC algorithms [62, 102], in that it includes both robustness and a short plan that does not necessarily reach the target.

The lines 2, 5, 8–13 are implemented in MATLAB. Before solving each subproblem in line 15, MATLAB forms the MILP constraints using both the static parameters such as vehicle dynamics limit, target location, and constraint tightening margin, and the dynamically updated parameters such as current vehicle states, obstacle boundaries, and other vehicles' plans. More details on the MILP implementation are shown in Appendix B. Then, the MILP solver CPLEX is invoked in line 15. The

optimized control states and inputs are extracted from CPLEX into MATLAB and is sent to the vehicle in line 17. Note that in the hardware experiment section 5.6, MATLAB receives the measured states from the vehicle in line 9.

5.4.3 Robust Feasibility

Even though each subproblem only uses local information, the robust feasibility of the entire fleet can be proven using an approach that parallels [110].

Theorem 5.2. *The system* (5.1) *controlled by Algorithm 5.2 satisfies all the local and coupling constraints* (5.3)–(5.4) *under the action of bounded disturbances* (5.2) *for all positive k, if feasible solutions to all subproblems $\mathsf{P}^1(\boldsymbol{x}_k^1), \ldots, \mathsf{P}^n(\boldsymbol{x}_k^n)$ can be found at time* 0.

Proof. The proof is based on a recursion and similar to the proof of Theorem 5.1 in Appendix 5.A. Without loss of generality, the planning order is assumed to be $1, 2, \ldots, n$. The proof considers two main stages, as outlined below. More details are found in Appendix 5.B.

0. Assume all the subproblems $\mathsf{P}^p(\boldsymbol{x}_k^p)$ have a feasible solution at time k.

1. Given feasible solutions of vehicles $1, \ldots, n$ at time k, it can be shown that a feasible solution exists to the first subproblem $\mathsf{P}^1(\boldsymbol{x}_{k+1}^1)$ at time $k+1$ for all disturbances $\boldsymbol{w}_k^1$ acting on the vehicle 1 despite the change in the neighbor set $\mathcal{I}_{k+1}^1$. This is done by showing that the following candidate solution is feasible.

$$\hat{\boldsymbol{u}}_{k+j+1|k+1}^1 = \boldsymbol{u}_{k+j+1|k}^1 + P_{j+1}^1 \boldsymbol{w}_k^1, \qquad \forall j^- \tag{5.29a}$$

$$\hat{\boldsymbol{x}}_{k+j+1|k+1}^1 = \boldsymbol{x}_{k+j+1|k}^1 + L_j^1 \boldsymbol{w}_k^1, \qquad \forall j \tag{5.29b}$$

$$\hat{\boldsymbol{u}}_{k+N|k+1}^1 = \kappa^1(\hat{\boldsymbol{x}}_{k+N|k+1}^1) \tag{5.29c}$$

$$\hat{\boldsymbol{x}}_{k+N+1|k+1}^1 = A^1 \hat{\boldsymbol{x}}_{k+N|k+1}^1 + B^1 \hat{\boldsymbol{u}}_{k+N|k+1}^1 \tag{5.29d}$$

$$\hat{\mathcal{S}}_{k+1}^1 = \mathcal{S}_k^1 \tag{5.29e}$$

2. Given feasible solutions of vehicle $1, \ldots, p$ at time $k+1$ and $p+1, \ldots, n$ at time k, it can be shown that a feasible solution exists to the next subproblem $\mathsf{P}^{p+1}(\boldsymbol{x}_{k+1}^{p+1})$, by showing the feasibility of a candidate sequence. Similar to (5.29) in Step 1, the candidate solution is constructed by shifting the previous plan for vehicle $p+1$, assumed known in Step 0, by one time step and adding a perturbation sequence using the predetermined controller P_j^{p+1}.

Therefore, at $k+1$, all subproblems $\mathsf{P}^1(\boldsymbol{x}_{k+1}^1), \ldots, \mathsf{P}^n(\boldsymbol{x}_{k+1}^n)$ are feasible. □

5.4.4 Remarks

Remark 5.2. *Simple Initialization*: Initializing this algorithm requires the other vehicles' previous solution, as shown in (5.26) and (5.27). However, a simple initialization technique such as loiter circle can be used, as discussed in Remark 5.1 of the RSBK algorithm.

Remark 5.3. *Scalability*: If each subproblem includes the interactions with all the other vehicles, as in [102], the number of constraints grows rapidly with the size of the fleet, which would increase the problem complexity. The algorithm presented here only requires the information about its neighbors, resulting in a more scalable approach. Furthermore, each vehicle only needs the information from its neighbors, so that the algorithm requires much less communication bandwidth.

5.4.5 Simultaneous Computation

This section removes the assumption on the strict ordering and enables simultaneous computation among vehicles.

Theorem 5.3. *Two vehicles p and q can generate trajectories simultaneously without causing infeasibility in the algorithm if $p \notin \mathcal{I}^q$ (and hence $q \notin \mathcal{I}^p$).*

Proof. By the definition of neighbor $\mathcal{I}^p$ and $\mathcal{I}^q$, the plans for p and q have no conflict. Given an arbitrary vehicle r ($\neq p, q$), both optimizations by p and q ensure that

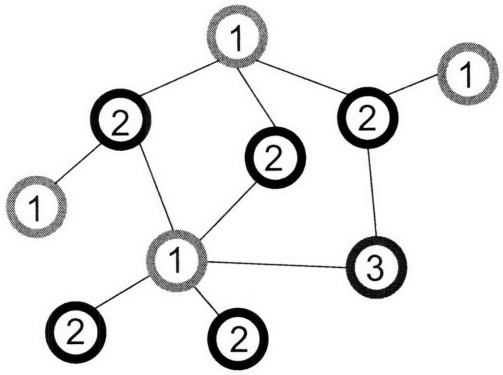


Figure 5-3: Output of Brelaz's heuristic algorithm for vertex coloring. Each node represents a vehicle, while each line connecting two nodes represents that they are neighborhood. The number is a group label for the vehicle and vehicles with the same group label can compute simultaneously.

the same candidate plan similar to (5.29) for each vehicle r is feasible. Thus, when p and q calculate simultaneously, any vehicle r has a feasible solution at the next optimization. □

By applying this theorem to pairs of vehicles in the fleet, it can be shown that more than two vehicles can perform optimization simultaneously. The vehicles that compute simultaneously are grouped together, and the number of vehicles that compute simultaneously is to be maximized. This grouping problem is cast as a vertex coloring problem on the vehicle graph $\mathcal{G}_k$, where each vertex represents a vehicle and vertices are connected if they are neighbors. The goal is to color all the vertices with a minimum number of colors while using different colors for adjacent vertices. Brelaz's heuristic algorithm [111] is used here because it provides good solutions very rapidly. Vehicles of the same color are in one group and can compute their solutions simultaneously.

Algorithm 5.3 shows a Brelaz's algorithm. This algorithm orders the color from 1 to n, where n is a number of vertices. The vertex degree is defined as the number of adjacent vertices, and the color degree is defined as the number of adjacent vertices that have already been colored. The input to this algorithm is a graph of n vertices and an adjacency matrix.

Figure 5-3 shows a simple example where Brelaz's algorithm is applied to a graph

Algorithm 5.3 Brelaz's algorithm

1: Initialize by setting all vertices uncolored
2: **while** there is an uncolored vertex **do**
3: Find an uncolored vertex with the highest color degree
4: **if** more than one vertices are found **then**
5: Choose the vertex of the largest degree
6: **end if**
7: Color the vertex with the smallest color that does not conflict with its neighbor's color
8: **end while**

of 10 vehicles. Note that in order to color the vehicles, the location of all the vehicles in the connected graph must be known. A central ground station can be introduced to run the grouping algorithm and determine the planning order. Alternatively, the vehicles can obtain this information by communicating only locally through neighbors. Then, the lines 5 and 7 of the DRSBK algorithm in Algorithm 5.2 are modified to the following.

5. Ground station receives vehicle positions $\boldsymbol{r}_k^p$, runs the grouping algorithm, and determines the planning order. Each vehicle updates the neighbor set $\mathcal{I}_k^p$.

7. For each group, do the following simultaneously for all vehicles p's in the group.

5.5 Simulation Results

5.5.1 Vehicle Model

A point-mass dynamics model is used to approximate the translational dynamics of UAVs

$$\begin{bmatrix} \boldsymbol{r}_{k+1}^p \\ \boldsymbol{v}_{k+1}^p \end{bmatrix} = A^p \begin{bmatrix} \boldsymbol{r}_k^p \\ \boldsymbol{v}_k^p \end{bmatrix} + B^p \boldsymbol{a}_k^p + \boldsymbol{w}_k^p$$

$$A^p = \begin{bmatrix} I_2 & \Delta t\, I_2 \\ O_2 & I_2 \end{bmatrix}, \quad B^p = \begin{bmatrix} \frac{(\Delta t)^2}{2} I_2 \\ \Delta t I_2 \end{bmatrix}$$

where $\boldsymbol{r}^p$, $\boldsymbol{v}^p$, and $\boldsymbol{a}^p$ are the position, the velocity, and the acceleration vector respectively. Matrices I_2 and O_2 express an identity matrix and a zero matrix of size 2 respectively. The disturbance $\boldsymbol{w}_k^p$ enters through the input acceleration and

$$\boldsymbol{w}_k^p \in \mathcal{W}^p = \left\{ \boldsymbol{w} \mid \boldsymbol{w} = B^p \boldsymbol{n},\ \boldsymbol{n} \in \mathbb{R}^2,\ \|\boldsymbol{n}\|_\infty \leq w_{\max} \right\}. \tag{5.30}$$

The local constraints include the obstacle avoidance, the maximum/minimum speed, and the maximum input constraints

$$\begin{aligned} \boldsymbol{r}_k^p &\notin \mathcal{O}, \\ v_{\min} \leq \|\boldsymbol{v}_k^p\|_2 &\leq v_{\max} \\ \|\boldsymbol{a}_k^p\|_2 &\leq a_{\max} \end{aligned}$$

where $\mathcal{O} \subset \mathbb{R}^2$ expresses the no-fly zones, and $v_{\min}, v_{\max}, a_{\max}$ are the minimum speed, maximum speed, and maximum acceleration of the vehicle. A two-step nilpotent controller K^p for this system is $K^p = \left[-\frac{1}{\Delta t^2} I_2,\ -\frac{3}{2\Delta t} I_2\right]$, which enables the use of nominal invariant set as a safety set. Obtaining P_j through $P_{j+1} = K^p L_j$ and performing constraint tightening (5.17b) give the following constraint set [102]

$$\boldsymbol{r}_{k+j|k}^p \notin \mathcal{O} \oplus \alpha_j \mathcal{B} \tag{5.31}$$

$$v_{\min} + \beta_j \leq \left\| \boldsymbol{v}_{k+j|k}^p \right\|_2 \leq v_{\max} - \beta_j \tag{5.32}$$

$$\left\| \boldsymbol{a}_{k+j|k}^p \right\|_2 \leq a_{\max} - \gamma_j \tag{5.33}$$

where constraint contraction parameters α, β, and γ are defined in (5.50) in Appendix 5.C. The set $\mathcal{B}$ represents a 2D unit box, i.e., $\mathcal{B} = \{x \in \mathbb{R}^2 \mid \|x\|_\infty \leq 1\}$. Note that (5.31) expands the no-fly zones to guarantee robust feasibility. The cost map calculation is based on the expanded obstacles $\mathcal{O} \oplus \alpha_{N-1}\mathcal{B}$. The inter-vehicle avoidance

constraints in p's optimization are written as

$$\begin{aligned} \left\| \boldsymbol{r}^p_{k+j|k} - \boldsymbol{r}^q_{k+j|k} \right\| &\geq 2d + 2\alpha_j, \qquad \text{if } \mathtt{ord}(q) < \mathtt{ord}(p),\ q \in \mathcal{I}^p_k \\ \left\| \boldsymbol{r}^p_{k+j|k} - \boldsymbol{r}^q_{k+j|k-1} \right\| &\geq 2d + \alpha_j + \alpha_{j+1}, \quad \text{if } \mathtt{ord}(q) > \mathtt{ord}(p),\ q \in \mathcal{I}^p_k \end{aligned}$$

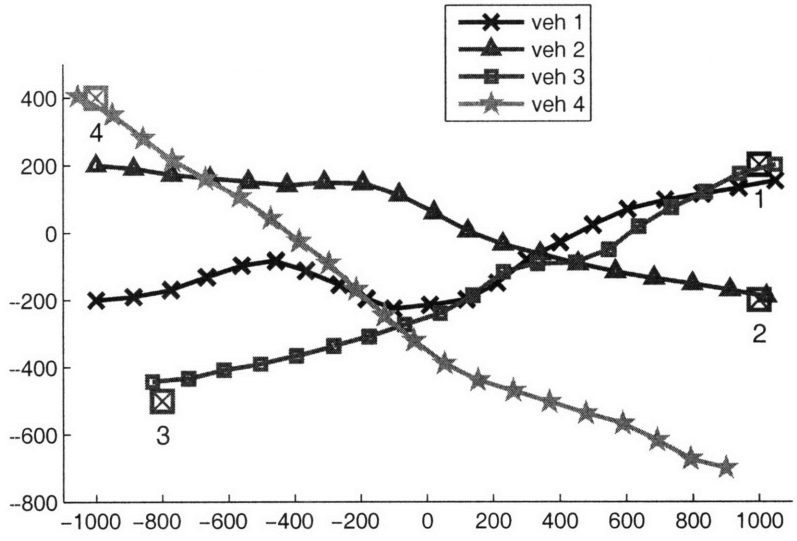
where $\boldsymbol{r}^q_{\cdot|\cdot}$ are sent from p's neighbors. The terminal safety sets $\mathcal{S}^p_k$ must not overlap with each other, as shown in (5.26), so that the sets $\mathcal{S}^q_k$ $(\forall q \neq p)$ are treated as no-fly zones after time step $k+N-1$ in the optimization $\mathsf{P}^p(\boldsymbol{x}^p_k)$. These non-convex avoidance constraints are implemented using MILP. More details are found in Appendix 5.C.

5.5.2 Multi-UAV Scenarios

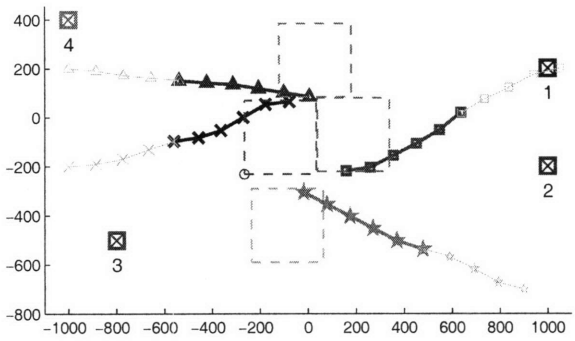
The simulations used homogeneous fixed-wing UAVs. The maneuver limit of the vehicle is given by $v_{\min} = 18\,\mathrm{m/s}$, $v_{\max} = 24\,\mathrm{m/s}$, $a_{\max} = 3.84\,\mathrm{m/s^2}$. The disturbance magnitude $w_{\max}$ is 5% of the control authority $a_{\max}$. The planning horizon length N is 5. Fixed-wing UAVs have minimum speed limit, and a safety loitering circle is used as a terminal invariant set [62]. For simplicity, in this section, the simultaneous computation is implemented as a sequential computation on a single computer but in the same simulation time step. The hardware experiments presented in Section 5.6 use the distributed implementation where each vehicle has one onboard processor.

The DRSBK algorithm was tested in the following two scenarios. The first scenario uses four vehicles with vehicle avoidance constraints. Figure 5-4(a) shows the entire trajectories. Goals are marked with ⊠ together with the corresponding vehicle indices. Figure 5-4(b) shows the plans made at time $k = 5$. The rectangle in dashed lines shows a safety region where the safety circle is contained and the other vehicles cannot enter after time $k + N$. Note that the plan of the vehicle 4 (marked with $\star$) aims for the corner (marked with $\circ$) of this rectangle of the vehicle 1 because this corner is in the cost map. As shown in Figure 5-4(c)–(e), it is acceptable for the plan of one vehicle to pass through the safety region for another. The terminal set (5.26) only requires that the safety regions do not overlap each other.

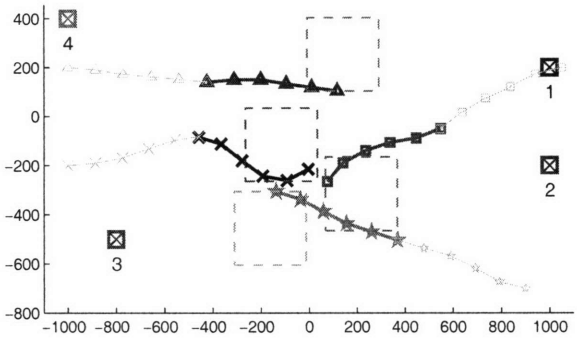
The second scenario is much more complicated and involves ten vehicles and four



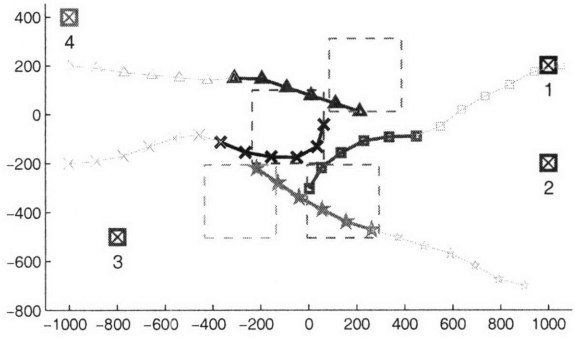
(a) Trajectories



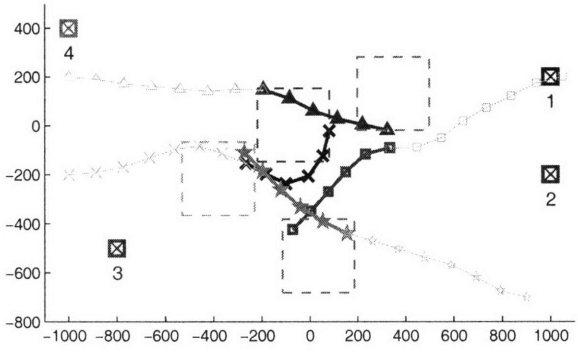
(b) Snapshot at time 5.



(c) Snapshot at time 6.

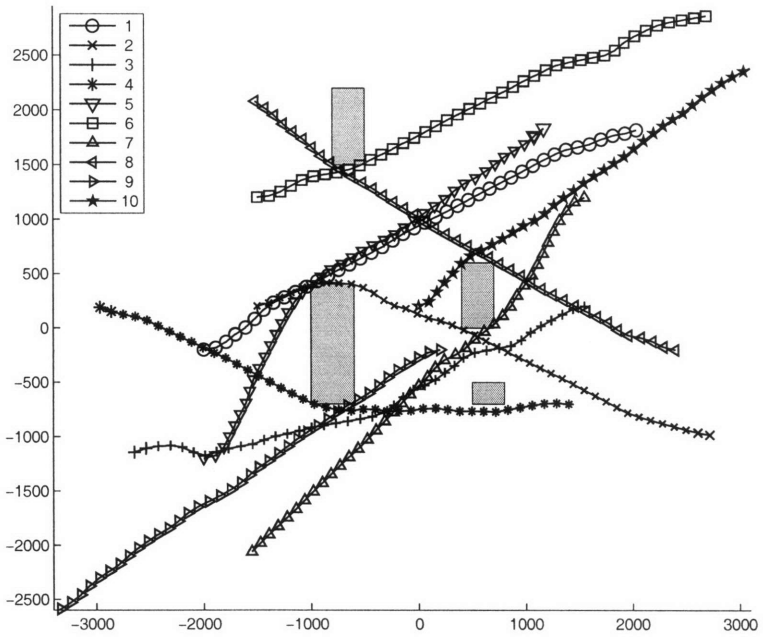


(d) Snapshot at time 7.

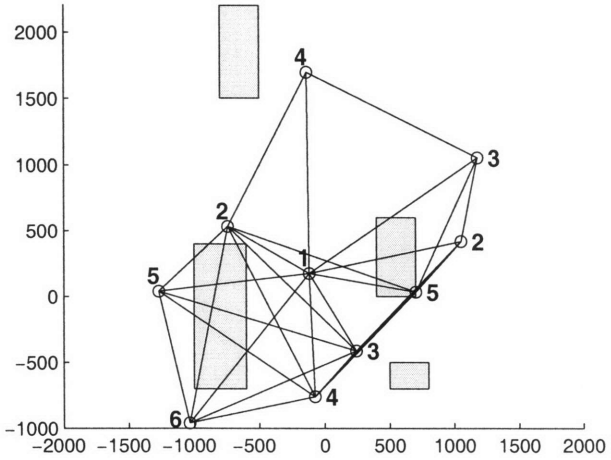


(e) Snapshot at time 8, showing the successful avoidance maneuvers.

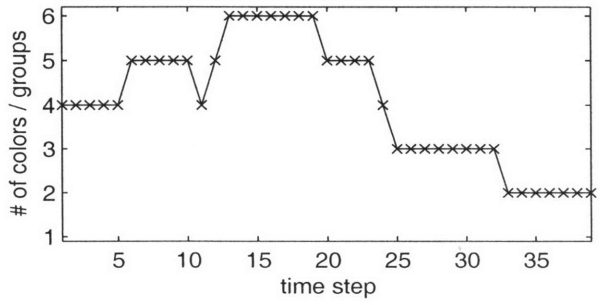
Figure 5-4: Trajectories generated by DRSBK in a four vehicle scenario. The goal points are shown with ⊠ with the corresponding vehicle index. Note that squares containing safety circles do not overlap with each other.



(a) Trajectories of all the vehicles



(b) Graph representation of neighborhood at time 14



(c) Time history of the vehicle grouping

Figure 5-5: Ten vehicle scenario with four obstacles.

Table 5.1: Average computation time (seconds) of each subproblem

Scenario	Cost map calculation	Optimization (MILP)
4 veh	0.04	0.21
10 veh (local comm.)	0.21	0.25
10 veh (full comm.)	0.21	0.37

obstacles. Figure 5-5(a) shows the trajectories of all ten vehicles. Although computation was done on one processor in this section, the grouping algorithm was included to investigate the potential for speed-up by simultaneous computation. Figure 5-5(b) shows a snapshot of the vehicle locations (marked with $\circ$) at time $t = 14$. The neighbors are connected by the lines and each vehicle is labeled with a color/group number. Note that no two vehicles connected to each other have the same group number. The vehicles in the same group can simultaneously solve their optimization without any conflict in their trajectories. Figure 5-5(c) shows the time history of the number of colors required for grouping the vehicles. The number of groups is low when the vehicles are far apart, but as might be expected, this increases to six in the middle of the mission when the vehicles are in close proximity.

Table 5.1 shows the average computation time for these scenarios. The cost map calculation was done in MATLAB, and the MILP optimization was solved using CPLEX 9.0 on Pentium IV 3.2GHz machine with 1GB of RAM. The computation time of the cost map calculation grows with the number of vehicles because a larger number of loiter circles means that more obstacles must be considered. In order to demonstrate the effect of using only the local information, the ten vehicle scenario is tested also with a case where each vehicle includes all other vehicles as neighbors with full communication. The last two rows of Table 5.1 illustrate that DRSBK with local communication solves the problem much faster. The output of the grouping algorithm is used to enable simultaneous computation, and the number of groups that must compute sequentially is 10 in the full communication case, as opposed to 6 in the local communication case. This indicates the local communication architecture reduces the fleet computation time further by 40% in this scenario, compared to the

approach using the fully sequential computation.

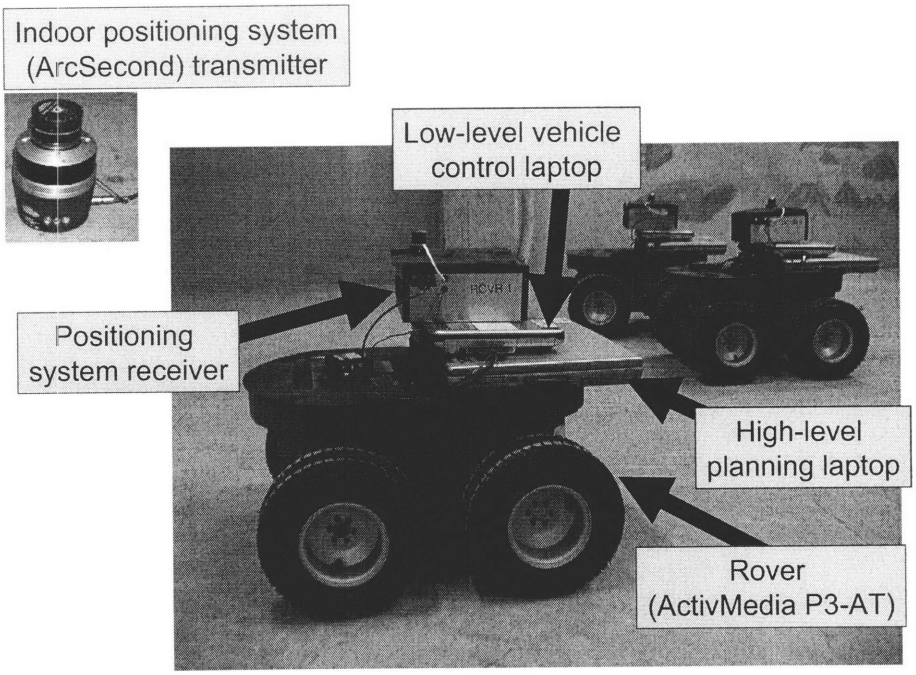
5.6 Experimental Results

This section presents experimental results of the DRSBK algorithm on the multi-rover testbed. The hardware demonstrations introduce realistic features such as computation and communication time delays and prediction errors that naturally arise from the various sources of uncertainty in the system, including the tracking errors from the low-level waypoint follower and modeling errors of the vehicle dynamics. These implementation challenges must be addressed by the algorithm in order to successfully generate trajectories online.

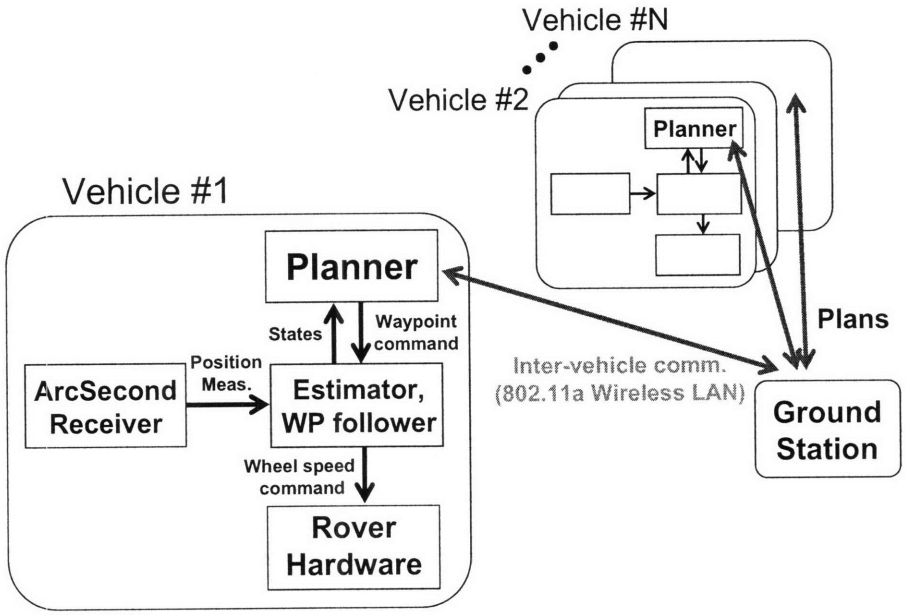
5.6.1 Testbed Setup

Figure 5-6 shows the testbed setup with the indoor positioning system from Arc-Second Constellation 3D-i and Pioneer 3-AT from ActivMedia Robotics. In order to demonstrate the online distributed computation amongst the vehicles in the fleet, each rover has two laptops, as shown in Fig. (a). A small "control" laptop performs the navigation and low-level vehicle control tasks, and a 2.4GHz "planning" laptop performs the DRSBK computation using a combination of MATLAB and CPLEX.

The control laptop runs an estimator for the position and the velocity estimate of the vehicle. For practical implementation, instead of applying the acceleration command $\boldsymbol{u}^*$ as in Algorithm step 17, the onboard planner sends the optimized trajectory to the control laptop, which generates wheel speed commands for the rover. A nonlinear guidance law [17] is used to implement a trajectory tracking controller, which runs at a faster rate than the DRSBK controller. This represents an apportionment of uncertainty in the problem, with the low-level handling fast dynamics and the high-level handling uncertainty in the environment, collision avoidance, and residual tracking errors. The ground station laptop runs a grouping algorithm at each time step, but all DRSBK calculations are done onboard, as shown in Figure 5-7. Each planning laptop communicates its local solution with its neighbors using the 802.11a



(a) Photos of the hardware



(b) Testbed architecture

Figure 5-6: Multi-rover testbed

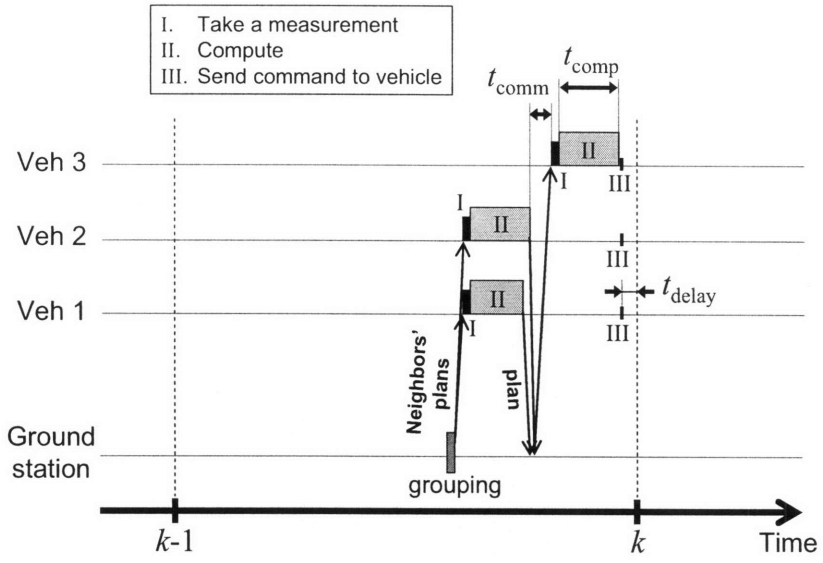


Figure 5-7: Timing of the DRSBK algorithm on the testbed.

wireless LAN. For this testbed, the inter-vehicle communication is facilitated using an access point connected through an Ethernet cable to the ground station laptop.

Figure 5-7 shows the timing of the experimental setup. This example has three vehicles in two groups where the vehicles 1 and 2 compute simultaneously. The control input of each vehicle is implemented using fixed discrete time steps. The planner takes a measurement (I) and propagates forward the measured states using the nominal model to predict the initial states $\boldsymbol{x}_k$ of the plan. This propagation compensates for the system delay that results from the computation time t_{comp}, the communication delay t_{comm}, and the actuation delay t_{delay}. It then computes the optimal control input (II) and waits until the control update time (III). The step size Δt between time step k and $k+1$ was 2.8 seconds for two-rover cases and 3.5 seconds for three-rover cases.

A typical experimental run starts by commanding the vehicles to drive straight in the initial heading direction. After 1.5 seconds, the first vehicle takes its measurement and the DRSBK loop starts. For other vehicles that have not made any plans, loiter circles starting from their current states are used as their initial feasible plan, as mentioned in Remark 5.2. This demonstrates the online initialization capability of this algorithm.

Given the applications of interest are multi-UAV coordination problems, the rovers have been modified to emulate the motion of a UAV in 2D. In particular, the vehicles are constrained to a maximum speed $v_{\max} = 0.25\,\mathrm{m/s}$, a minimum speed $v_{\min} = 0.044\,\mathrm{m/s}$, and a minimum turning radius $r_{\min} = 0.9\,\mathrm{m}$. The vehicle size is $d = 0.25\,\mathrm{m}$. The planning horizon length is three steps. The disturbance $\boldsymbol{w}_k^p$ is assumed to enter into position and velocity separately,

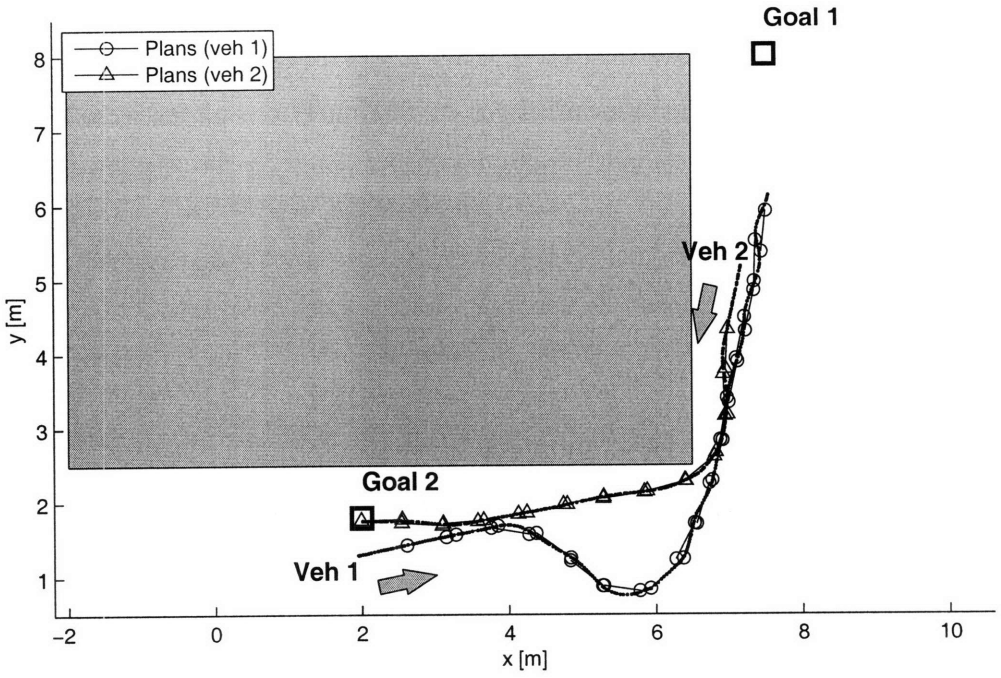
$$\boldsymbol{w}_k^p \in \mathcal{W} = \left\{ \boldsymbol{w} \in \mathbb{R}^4 \;\middle|\; \left\| [I_2,\ O_2]\boldsymbol{w} \right\|_2 \leq w_{r_{\max}},\ \left\| [O_2,\ I_2]\boldsymbol{w} \right\|_2 \leq w_{v_{\max}} \right\}. \tag{5.34}$$

Extensive testing of the vehicle on different types of flooring indicated that the prediction errors due to the uncertain vehicle dynamics, navigation errors and external disturbances are approximately $w_{r_{\max}} = 0.15\,\mathrm{m}$ and $w_{v_{\max}} = 0.05\,\mathrm{m/s}$. Due to the tightened constraints, the speed is constrained to be $0.14\,\mathrm{m/s} \leq v \leq 0.15\,\mathrm{m/s}$ after $N = 3$ steps.

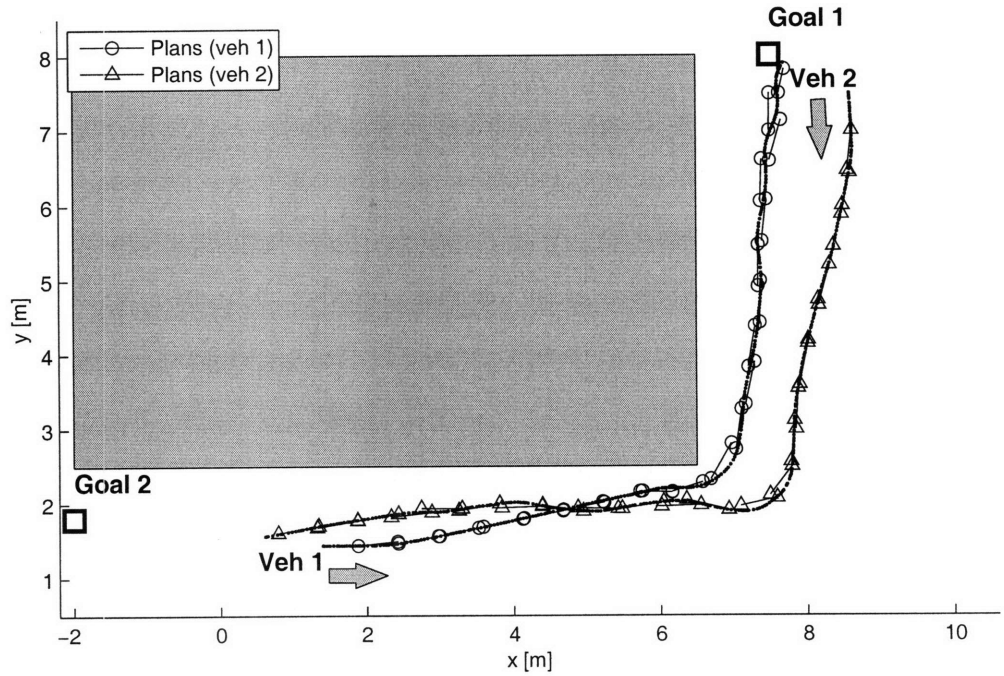
5.6.2 Results

Scenarios are constructed to highlight several features of DRSBK algorithm: onboard laptops generate trajectories online, which shows the computational advantages for real-time applications; the vehicles are required to maneuver in a constrained environment, which demonstrates the robust feasibility under the action of disturbances; plans based on distributed computation can satisfy the coupling collision avoidance constraints.

Test 1: The first set of experiments was designed to test the obstacle and vehicle avoidance using two rovers. During the first few steps in each run, the separation between the two vehicles was more than $2D = 6.34$ m, and the onboard computers optimize trajectories simultaneously. However, as they move towards each other, the planning horizons overlap, and they compute the solutions sequentially. For the purposes of the demonstration, the experiment is terminated once the vehicle avoidance and obstacle avoidance maneuvers are completed. Figure 5-8 and Figure 5-9 show four runs performed on this testbed. DRSBK algorithm maintained feasibility

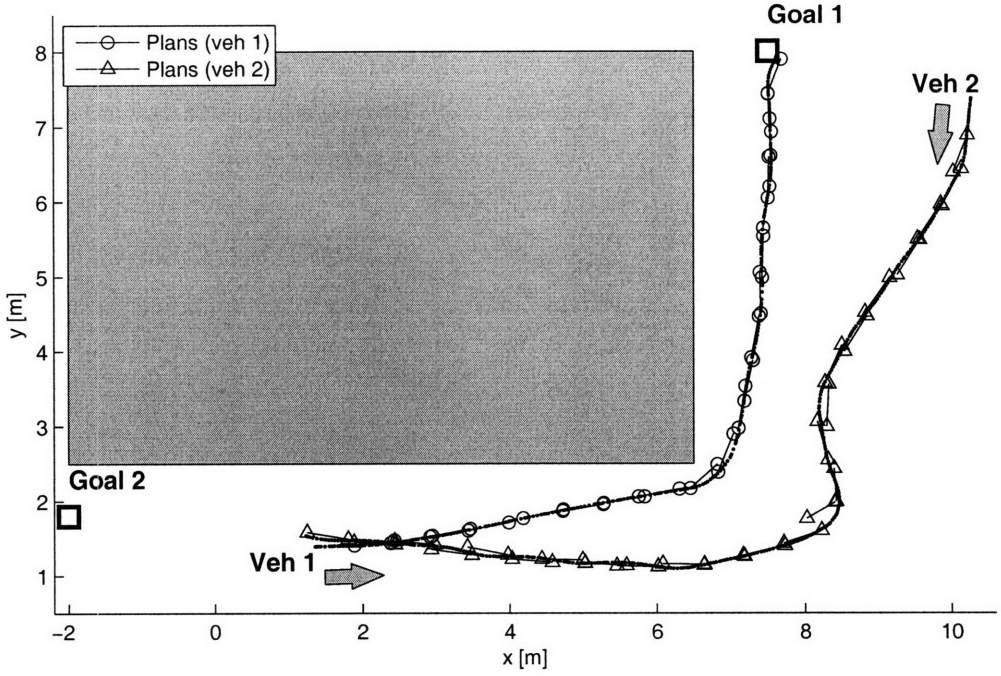


(a) Run #1

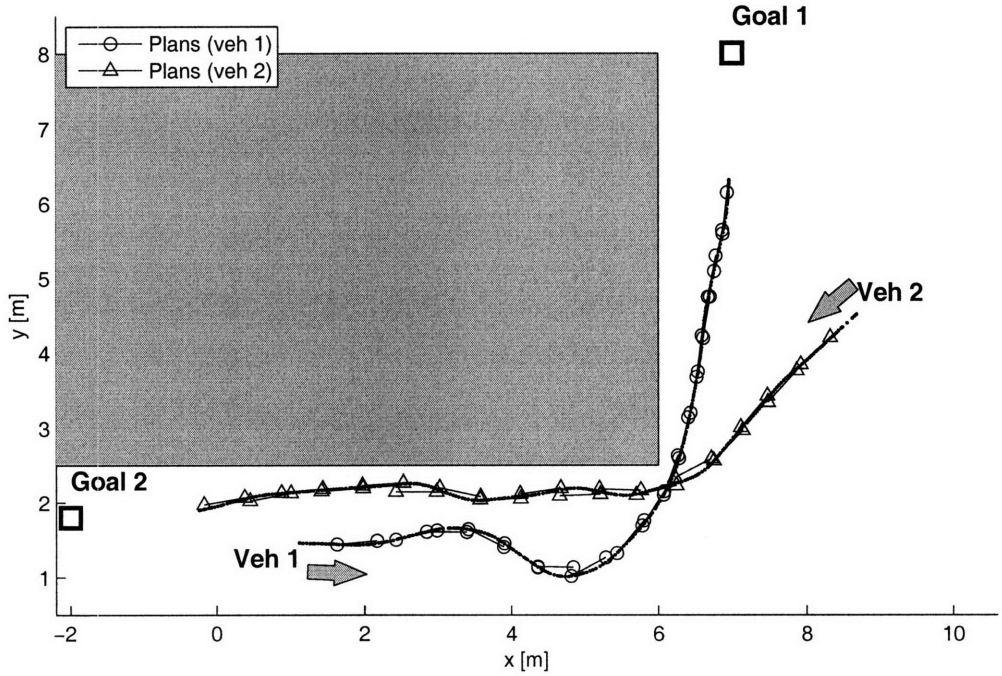


(b) Run #2

Figure 5-8: Two vehicle experiment results. The arrows show the initial heading directions of the vehicles. The goals are marked with □.

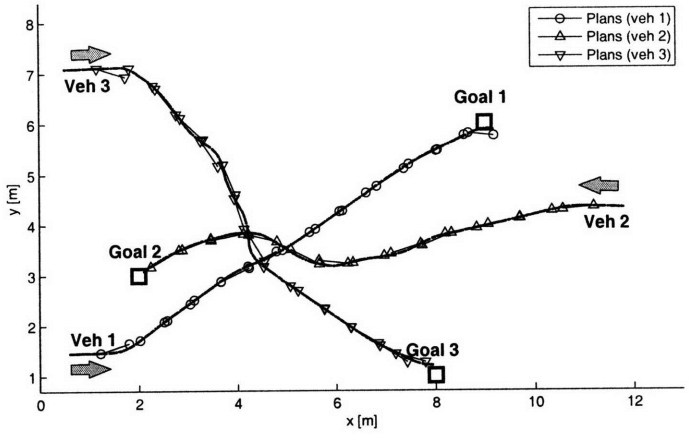


(a) Run #3

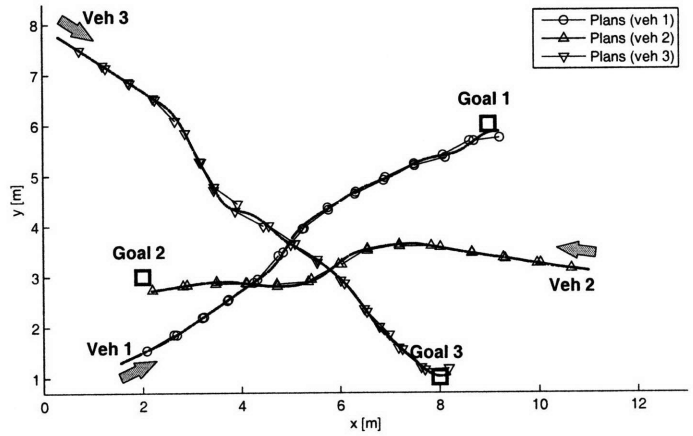


(b) Run #4

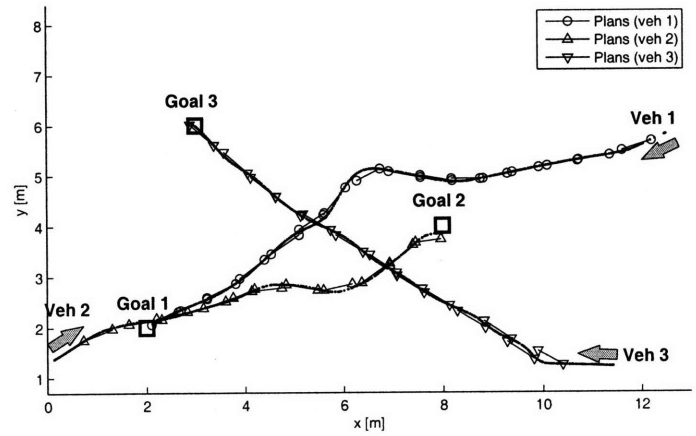
Figure 5-9: Two vehicle experiment results. The arrows show the initial heading directions of the vehicles. The goals are marked with □.



(a) Run #1



(b) Run #2



(c) Run #3

Figure 5-10: Three vehicle experiment results. The arrows show the initial heading directions of the vehicles. The goals are marked with □.

under the action of the disturbances, and all runs show the robust vehicle avoidance and obstacle avoidance based on the online distributed trajectory generation.

Test 2: The second set of runs examines vehicle avoidance maneuvers using three rovers that are forced to execute a crossing pattern. Figure 5-10 presents the executed trajectories for three runs with different initial locations and headings. Note that the resolution strategies differ with the scenario. One of the key features of MILP is that it handles the non-convexity directly and looks for solutions on all sides of avoidance zones and conflicts. This example illustrates that DRSBK is making use of this functionality, as opposed to other methods that could simply refine the initial guess that are given. In these scenarios, vehicles 1 and 3 are initially neighbors because they are closer than $2D = 6.54\,\mathrm{m}$, and thus compute sequentially. Vehicle 2 is initially independent of that pair, and thus solves for its plan simultaneously with one of them. However, since the vehicles are crossing, vehicle 2 joins the pair after a few time steps, and then all three vehicles compute sequentially. Once the vehicles finish the avoidance maneuver near the middle of the figure, the group breaks up as the vehicles move apart and starts solving for the plans simultaneously again. The results demonstrate online dynamic grouping and re-grouping of the vehicles using the algorithm in Section 5.4.5.

5.7 Summary

This chapter presented a new distributed robust Model Predictive Control algorithm for multi-vehicle trajectory optimization. Each vehicle generates its control inputs by solving a subproblem in sequence, while freezing the plans of other vehicles. The solution is then communicated to other vehicles in the fleet. The approach extends previous results to ensure robust feasibility without having to plan all of the way to the goal and with only communicating the plans within a local neighborhood rather than the entire fleet. This two new features greatly reduce the computation effort and facilitate a significantly more general implementation architecture for the distributed trajectory optimization. Experimental results on a multi-vehicle testbed demonstrate many advantages of this algorithm including online distributed optimization, simultaneous computation, and the robust feasibility against the disturbances in the real environment.

5.A Proof of Theorem 5.1

If the optimization (5.8)–(5.16) is feasible at time k, then the vehicles satisfy all the constraints at time k. This is because the constraints (5.11) and (5.13) in the optimization ensure that the constraints (5.3) and (5.4) of the real system are satisfied through (5.17a), (5.19a), (5.9), and $\boldsymbol{u}^p_{k|k} = \boldsymbol{u}^p_k$.

Therefore, the proof needs to show that the optimization (5.8)–(5.16) is always feasible under the action of the bounded disturbance (5.2). The proof is based on a recursion that is similar to the proof of Theorem 3.1.

Candidate Solution at Time $k+1$

First, assume the form of the candidate solution at time $(k+1)$ as

$$\forall p: \quad \hat{\boldsymbol{u}}^p_{k+j+1|k+1} = \boldsymbol{u}^p_{k+j+1|k} + P^p_{j+1}\boldsymbol{w}^p_k, \quad \forall j^- \tag{5.35a}$$

$$\hat{\boldsymbol{x}}^p_{k+j+1|k+1} = \boldsymbol{x}^p_{k+j+1|k} + L^p_j\boldsymbol{w}^p_k, \quad \forall j \tag{5.35b}$$

$$\hat{\boldsymbol{u}}^p_{k+N|k+1} = \kappa^p(\hat{\boldsymbol{x}}^p_{k+N|k+1}) \tag{5.35c}$$

$$\hat{\boldsymbol{x}}^p_{k+N+1|k+1} = A^p\hat{\boldsymbol{x}}^p_{k+N|k+1} + B^p\hat{\boldsymbol{u}}^p_{k+N|k+1} \tag{5.35d}$$

$$\hat{\mathcal{R}}^p_{k+1} = \mathcal{R}^p_k \tag{5.35e}$$

$$\hat{\mathcal{S}}^p_{k+1} = \mathcal{S}^p_k \tag{5.35f}$$

which is constructed from the solution obtained at time k. Note that the disturbance realization $\boldsymbol{w}^p_k$ at time k is available at time $(k+1)$.

Initial Condition (5.9)

By setting $j = 0$ in (5.35b),

$$\begin{aligned}\hat{\boldsymbol{x}}^p_{k+1|k+1} &= \boldsymbol{x}^p_{k+1|k} + \boldsymbol{w}^p_k \\ &= A\boldsymbol{x}^p_{k|k} + B\boldsymbol{u}^p_{k|k} + \boldsymbol{w}^p_k \\ &= A\boldsymbol{x}^p_k + B\boldsymbol{u}^p_k + \boldsymbol{w}^p_k = \boldsymbol{x}^p_{k+1}\end{aligned}$$

Thus, the candidate solution satisfies the initial condition (5.9) at time $k+1$.

State Equation (5.10)

At time $k+1$, the state equation (5.10) for the prediction step $j = N-1$ is satisfied by the definition of $\hat{\boldsymbol{x}}^p_{k+N+1|k+1}$ in (5.35d). For $j = 0, \ldots, N-2$, from the definition of L^p_j (5.18),

$$L^p_{j+1}\boldsymbol{w}^p_k = A^p L^p_j \boldsymbol{w}^p_k + B^p P^p_{j+1}\boldsymbol{w}^p_k,$$

which holds for any $\boldsymbol{w}^p_k$. By adding this to the following state equation at time k

$$\boldsymbol{x}^p_{k+j+2|k} = A^p \boldsymbol{x}^p_{k+j+1|k} + B^p \boldsymbol{u}^p_{k+j+1|k}$$

and using (5.35a) and (5.35b), we have

$$\hat{\boldsymbol{x}}^p_{k+j+2|k+1} = A^p \hat{\boldsymbol{x}}^p_{k+j+1|k+1} + B^p \hat{\boldsymbol{u}}^p_{k+j+1|k+1}$$

for all p.

Local Constraints (5.11)

For prediction steps $j = 0, \ldots, N-2$,

$$\begin{aligned}
\hat{\boldsymbol{y}}^p_{k+j+1|k+1} &= C^p \hat{\boldsymbol{x}}^p_{k+j+1|k+1} + D^p \hat{\boldsymbol{u}}^p_{k+j+1|k+1} \\
&= C^p \boldsymbol{x}^p_{k+j+1|k} + D^p \boldsymbol{u}^p_{k+j+1|k} + C^p L^p_j \boldsymbol{w}^p_k + D^p P^p_{j+1} \boldsymbol{w}^p_k \\
&= \boldsymbol{y}^p_{k+j+1|k} + \left(C^p L^p_j + D^p P^p_{j+1}\right)\boldsymbol{w}^p_k.
\end{aligned}$$

Note that the solution at time k satisfies $\boldsymbol{y}^p_{k+j+1|k} \in \mathcal{Y}^p_{j+1}$, and the bounded disturbance is $\left(C^p L^p_j + D^p P^p_{j+1}\right)\boldsymbol{w}^p_k \in \left(C^p L^p_j + D^p P^p_{j+1}\right)\mathcal{W}^p$. Therefore, using the relation (5.17b), we have

$$\hat{\boldsymbol{y}}^p_{k+j+1|k+1} \in \mathcal{Y}^p_j, \qquad \forall \boldsymbol{w}^p_k.$$

Coupling Constraints (5.12)–(5.13)

For each prediction step $\forall j$,

$$\begin{aligned}\hat{\boldsymbol{z}}^p_{k+j+1|k+1,c} &= E^p_c \hat{\boldsymbol{x}}^p_{k+j+1|k+1} \\ &= E^p_c \boldsymbol{x}^p_{k+j+1|k} + E^p_c L^p_j \boldsymbol{w}^p_k \end{aligned} \tag{5.36}$$

Note that the solution at time k satisfies

$$\begin{aligned}\sum_{p=1}^{n} E^p_c \boldsymbol{x}^p_{k+j+1|k} &= \sum_{p=1}^{n} \boldsymbol{z}^p_{k+j+1|k,c} \\ &\in \mathcal{Z}_{j+1,c} \end{aligned} \tag{5.37}$$

and the disturbance terms are

$$\begin{aligned} E^p_c L^p_j \boldsymbol{w}^p_k &\in E^p_c L^p_j \mathcal{W}^p, \quad \forall p \\ \sum_{p=1}^{n} E^p_c L^p_j \boldsymbol{w}^p_k &\in \Big(E^1_c L^1_j \mathcal{W}_1 \oplus \ldots \oplus E^n_c L^n_j \mathcal{W}_n \Big). \end{aligned} \tag{5.38}$$

Thus, using (5.37), (5.38), and the relation (5.19b), the summation of (5.36) over the vehicles is

$$\begin{aligned}\sum_{p=1}^{n} \hat{\boldsymbol{z}}^p_{k+j+1|k+1,c} &\in \mathcal{Z}_{j+1,c} \oplus \Big(E^1_c L^1_j \mathcal{W}_1 \oplus \ldots \oplus E^n_c L^n_j \mathcal{W}_n \Big) \\ &\subseteq \mathcal{Z}_{j,c}. \end{aligned}$$

Note that the Pontryagin difference has the following property

$$(\mathcal{A} \sim \mathcal{B}) \oplus \mathcal{B} \subseteq \mathcal{A} \tag{5.39}$$

and $(\mathcal{A} \sim \mathcal{B}) \oplus \mathcal{B} \neq \mathcal{A}$ in general [95].

Terminal Constraints (5.14)–(5.16)

Check if the terminal step of the candidate solution satisfies the terminal constraint $\hat{\boldsymbol{x}}^p_{k+N+1|k+1} \in \hat{\mathcal{S}}^p_{k+1}$. By letting $j = N-1$ in (5.35b),

$$\hat{\boldsymbol{x}}^p_{k+N|k+1} = \boldsymbol{x}^p_{k+N|k} + L^p_{N-1}\boldsymbol{w}^p_k.$$

Since

$$\begin{aligned} \boldsymbol{x}^p_{k+N|k} &\in \mathcal{S}^p_k = \mathcal{R}^p_k \sim L^p_{N-1}\mathcal{W}^p \\ L^p_{N-1}\boldsymbol{w}^p_k &\in L^p_{N-1}\mathcal{W}^p \end{aligned}$$

we get

$$\hat{\boldsymbol{x}}^p_{k+N|k+1} \in \mathcal{R}^p_k = \hat{\mathcal{R}}^p_{k+1}.$$

Using the property (5.16),

$$\begin{aligned} &\begin{cases} A^p\hat{\boldsymbol{x}}^p_{k+N|k+1} + B^p\kappa^p\big(\hat{\boldsymbol{x}}^p_{k+N|k+1}\big) + L^p_{N-1}\boldsymbol{w}^p \in \mathcal{R}^p_k, \quad \forall \boldsymbol{w}^p \in \mathcal{W}^p \\ C^p\hat{\boldsymbol{x}}^p_{k+N|k+1} + D^p\kappa^p\big(\hat{\boldsymbol{x}}^p_{k+N|k+1}\big) \in \mathcal{Y}^p_{N-1} \end{cases} \\ \Rightarrow\ &\begin{cases} \hat{\boldsymbol{x}}^p_{k+N+1|k+1} + L^p_{N-1}\boldsymbol{w}^p_k \in \mathcal{R}^p_k, \quad \forall \boldsymbol{w}^p_k \in \mathcal{W}^p \\ C^p\hat{\boldsymbol{x}}^p_{k+N|k+1} + D^p\hat{\boldsymbol{u}}^p_{k+N|k+1} \in \mathcal{Y}^p_{N-1} \end{cases} \\ \Rightarrow\ &\begin{cases} \hat{\boldsymbol{x}}^p_{k+N+1|k+1} \in \mathcal{R}^p_k \sim L^p_{N-1}\mathcal{W}^p = \mathcal{S}^p_k = \hat{\mathcal{S}}^p_{k+1} \\ C^p\hat{\boldsymbol{x}}^p_{k+N|k+1} + D^p\hat{\boldsymbol{u}}^p_{k+N|k+1} \in \mathcal{Y}^p_{N-1} \end{cases} \end{aligned}$$

Thus, the candidate solution also satisfies the terminal constraint (5.14). The last line shows that the local constraint (5.11) for prediction step $j = N-1$ is satisfied at time $k+1$. Since the safety sets $\mathcal{S}^1_k, \ldots, \mathcal{S}^n_k$ at time k satisfies the property (5.15)–(5.16), the candidate safety sets $\hat{\mathcal{S}}^1_{k+1}, \ldots, \hat{\mathcal{S}}^n_{k+1}$ given by (5.35f) also satisfy (5.15)–(5.16).

Therefore, under the action of a bounded disturbance $\boldsymbol{w}^p_k \in \mathcal{W}^p$, the candidate solution (5.35) satisfies all the state prediction equation, local constraints $\mathcal{Y}^p_j$, and coupling constraints $\mathcal{Z}_{j,c}$ at time $k+1$, and the terminal state $\hat{\boldsymbol{x}}^p_{k+N+1|k+1}$ lies in the

safety set $\hat{\mathcal{S}}^p_{k+1} = \mathcal{S}^p_k$, which has the robust invariance property of $\mathcal{R}^p_k$. □

5.B Proof of Theorem 5.2

As stated in the outline of the proof on p.126, the proof needs to show the following two arguments.

1. Given feasible solutions of vehicles $1, \ldots, n$ at time k, a feasible solution exists to the first subproblem $\mathsf{P}^1(\boldsymbol{x}^1_{k+1})$ at time $k+1$.

2. Given feasible solutions of vehicle $1, \ldots, p$ at time $k+1$ and $p+1, \ldots, n$ at time k, a feasible solution exists to the next subproblem $\mathsf{P}^{p+1}(\boldsymbol{x}^{p+1}_{k+1})$ at time $k+1$.

5.B.1 Feasibility of Vehicle 1

All the vehicles $1, \ldots, n$ are assumed to have a feasible solution at time k, satisfying the following constraints.

$$
\begin{aligned}
\forall p: \quad \boldsymbol{x}^p_{k+j+1|k} &= A^p \boldsymbol{x}^p_{k+j|k} + B^p \boldsymbol{u}^p_{k+j|k} \\
\boldsymbol{y}^p_{k+j|k} &= C^p \boldsymbol{x}^p_{k+j|k} + D^p \boldsymbol{u}^p_{k+j|k} \in \mathcal{Y}^p_j \\
\boldsymbol{z}^p_{k+j|k,c} &= E^p_c \boldsymbol{x}^p_{k+j|k} \\
\boldsymbol{z}^p_{k+j|k,c} + \tilde{\boldsymbol{z}}^p_{k+j|k,c} &\in \mathcal{Z}^p_{j,c}, \quad \forall c \in \mathcal{C}^p_k \\
\boldsymbol{x}^p_{k+N|k} &\in \mathcal{S}^p_k \\
\forall \boldsymbol{x}^p \in \mathcal{S}^p_k \;\Rightarrow\; &\begin{cases}
A^p \boldsymbol{x}^p + B^p \kappa^p(\boldsymbol{x}^p) \in \mathcal{S}^p_k \\
C^p \boldsymbol{x}^p + D^p \kappa^p(\boldsymbol{x}^p) \in \mathcal{Y}^p_{N-1} \\
\forall c \in \mathcal{C}^p_k : \sum_{q \in \mathcal{I}^p_k} E^q_c \boldsymbol{x}^q \in \mathcal{Z}_{N-1,c} \\
\qquad \forall (\boldsymbol{x}^{p_0}, \ldots, \boldsymbol{x}^{p_n}) \in \{\mathcal{S}^{p_0}_k \times \ldots \times \mathcal{S}^{p_n}_k\}
\end{cases}
\end{aligned}
$$

We assume the following form of the candidate solution

$$\hat{\boldsymbol{u}}^1_{k+j+1|k+1} = \boldsymbol{u}^1_{k+j+1|k} + P^1_{j+1}\boldsymbol{w}^1_k, \quad \forall j^- \tag{5.40a}$$

$$\hat{\boldsymbol{x}}^1_{k+j+1|k+1} = \boldsymbol{x}^1_{k+j+1|k} + L^1_j\boldsymbol{w}^1_k, \quad \forall j \tag{5.40b}$$

$$\hat{\boldsymbol{u}}^1_{k+N|k+1} = \kappa^1(\hat{\boldsymbol{x}}^1_{k+N|k+1}) \tag{5.40c}$$

$$\hat{\boldsymbol{x}}^1_{k+N+1|k+1} = A^1\hat{\boldsymbol{x}}^1_{k+N|k+1} + B^1\hat{\boldsymbol{u}}^1_{k+N|k+1} \tag{5.40d}$$

$$\hat{\mathcal{S}}^1_{k+1} = \mathcal{S}^1_k \tag{5.40e}$$

and check if this satisfies all the constraints.

The initial condition (5.9), state equation (5.10), and local constraints (5.11) are identical to those in Appendix 5.A, and therefore satisfied by the candidate solution (5.40).

Coupling Constraints (5.25)

Since vehicle 1 is the first vehicle to plan at time $k+1$, other vehicles' plans (5.27) are

$$\tilde{\boldsymbol{z}}^1_{k+j+1|k+1,c} = \sum_{q\in\mathcal{I}^1_{k+1}, q\neq 1} \boldsymbol{z}^q_{k+j+1|k,c}\,. \tag{5.41}$$

Because vehicle n has a feasible solution at time k,

$$\sum_{q\in\mathcal{I}^n_k} \boldsymbol{z}^q_{k+j+1|k,c} \in \mathcal{Z}^n_{j,c}. \qquad \forall c \in \mathcal{C}^n_k\,. \tag{5.42}$$

The definition of neighbors and the coupling constraint set (5.23) ensure that other coupling constraints $c \notin \mathcal{C}^n_k$ are already satisfied prior to the optimization by n. By combining this with (5.42), the constraint satisfaction of the fleet is guaranteed

$$\sum_{q=1}^{n} \boldsymbol{z}^q_{k+j+1|k,c} \in \mathcal{Z}^n_{j,c} \qquad \forall c \tag{5.43}$$

By using (5.12) and (5.40b), $\forall j$,

$$\begin{aligned}\hat{\boldsymbol{z}}^1_{k+j+1|k+1,c} &= E^1_c \hat{\boldsymbol{x}}^1_{k+j+1|k+1} \\ &= E^1_c \boldsymbol{x}^1_{k+j+1|k} + E^1_c L^1_j \boldsymbol{w}^1_k.\end{aligned}$$

Combining above equations, we have

$$\begin{aligned}\hat{\boldsymbol{z}}^1_{k+j+1|k+1,c} + \tilde{\boldsymbol{z}}^1_{k+j+1|k+1,c} &= \boldsymbol{z}^1_{k+j+1|k,c} + E^1_c L^1_j \boldsymbol{w}^1_k + \sum_{q=2}^{n} \boldsymbol{z}^q_{k+j+1|k,c} \\ &\in \mathcal{Z}^n_{j+1,c} \oplus E^1_c L^1_j \mathcal{W}_1 \\ &\subseteq \mathcal{Z}^1_{j,c}, \qquad \forall c.\end{aligned}$$

The last step used the definition of the set $\mathcal{Z}$ in (5.28c) and the property of the Pontryagin difference (5.39).

Terminal Constraints (5.14) and (5.26)

Check if the terminal step of the candidate solution satisfies the terminal constraint $\hat{\boldsymbol{x}}^1_{k+N+1|k+1} \in \hat{\mathcal{S}}^1_{k+1}$. Since DRSBK uses a nilpotent policy $L^p_{N-1} = 0$, we have the following by letting $j = N-1$ in (5.40b).

$$\hat{\boldsymbol{x}}^1_{k+N|k+1} = \boldsymbol{x}^1_{k+N|k}$$

Since the solution from the previous time step k ensures $\boldsymbol{x}^1_{k+N|k} \in \mathcal{S}^1_k$, the property (5.26) gives

$$\begin{cases} A^1 \hat{\boldsymbol{x}}^1_{k+N|k+1} + B^1 \kappa^1\big(\hat{\boldsymbol{x}}^1_{k+N|k+1}\big) \in \mathcal{S}^1_k \\ C^1 \hat{\boldsymbol{x}}^1_{k+N|k+1} + D^1 \kappa^1\big(\hat{\boldsymbol{x}}^1_{k+N|k+1}\big) \in \mathcal{Y}^1_{N-1} \\ \qquad \forall c \in \mathcal{C}^1_k : \quad \sum_{q \in \mathcal{I}^1_k} E^q_c \boldsymbol{x}^q \in \mathcal{Z}_{N-1,c} \\ \qquad\qquad \forall (\boldsymbol{x}^{p_0}, \ldots, \boldsymbol{x}^{p_n}) \in \{\mathcal{S}^{p_0}_k \times \ldots \times \mathcal{S}^{p_n}_k\} \end{cases}$$

$$\Rightarrow \quad \begin{cases} \hat{\boldsymbol{x}}^1_{k+N+1|k+1} \in \mathcal{S}^1_k = \hat{\mathcal{S}}^1_{k+1} \\ C^1 \hat{\boldsymbol{x}}^1_{k+N|k+1} + D^1 \hat{\boldsymbol{u}}^1_{k+N|k+1} \in \mathcal{Y}^1_{N-1} \end{cases}$$

Thus, the candidate solution also satisfies the terminal constraint (5.14) for the optimization at time $k+1$. The last line shows that the local constraint (5.11) for prediction step $j = N-1$ is also satisfied at time $k+1$. Since the safety sets $\mathcal{S}^1_k$ at time k satisfies the property (5.26), the candidate safety sets $\hat{\mathcal{S}}^1_{k+1}$ given by (5.40e) also satisfy (5.26).

5.B.2 Feasibility of Vehicle $p+1$

This subsection assumes the same form of the candidate solution for vehicle $p+1$

$$\hat{\boldsymbol{u}}^{p+1}_{k+j+1|k+1} = \boldsymbol{u}^{p+1}_{k+j+1|k} + P^{p+1}_{j+1} \boldsymbol{w}^{p+1}_k, \quad \forall j^- \tag{5.44a}$$

$$\hat{\boldsymbol{x}}^{p+1}_{k+j+1|k+1} = \boldsymbol{x}^{p+1}_{k+j+1|k} + L^{p+1}_j \boldsymbol{w}^{p+1}_k, \quad \forall j \tag{5.44b}$$

$$\hat{\boldsymbol{u}}^{p+1}_{k+N|k+1} = \kappa^{p+1}(\hat{\boldsymbol{x}}^{p+1}_{k+N|k+1}) \tag{5.44c}$$

$$\hat{\boldsymbol{x}}^{p+1}_{k+N+1|k+1} = A^1 \hat{\boldsymbol{x}}^{p+1}_{k+N|k+1} + B^1 \hat{\boldsymbol{u}}^{p+1}_{k+N|k+1} \tag{5.44d}$$

$$\hat{\mathcal{S}}^{p+1}_{k+1} = \mathcal{S}^{p+1}_k \tag{5.44e}$$

and check if this satisfies all the constraints. Again, the initial condition (5.9), state equation (5.10), and local constraints (5.11) are identical to those in Appendix 5.A, and the candidate solution (5.44) satisfies them. The proof for the terminal constraints (5.14) and (5.26) is the same as the one given in Section 5.B.1 with a change of the superscript from 1 to $p+1$.

Coupling Constraints (5.25)

The term $\tilde{\boldsymbol{z}}^{p+1}_{k+j+1|k+1,c}$, as defined in (5.27), is

$$\tilde{\boldsymbol{z}}^{p+1}_{k+j+1|k+1,c} = \sum_{\substack{q \in \mathcal{I}^{p+1}_{k+1}, \\ q<p+1}} \boldsymbol{z}^q_{k+j+1|k+1,c} + \sum_{\substack{q \in \mathcal{I}^{p+1}_{k+1}, \\ q>p+1}} \boldsymbol{z}^q_{k+j+1|k,c}. \tag{5.45}$$

Because vehicle p has a feasible solution at time $k+1$,

$$\sum_{\substack{q\in\mathcal{I}^p_{k+1}\\ q\le p}} \boldsymbol{z}^q_{k+j+1|k+1,c} + \sum_{\substack{q\in\mathcal{I}^p_{k+1}\\ q>p}} \boldsymbol{z}^q_{k+j+1|k,c} \in \mathcal{Z}^p_{j,c}, \qquad \forall c \in \mathcal{C}^p_{k+1}.$$

The definition of neighbors and the coupling constraint set (5.23) ensure that other coupling constraints $c \notin \mathcal{C}^p_{k+1}$ are already satisfied prior to the optimization by p. Therefore,

$$\sum_{q=1}^{p} \boldsymbol{z}^q_{k+j+1|k+1,c} + \sum_{q=p+1}^{n} \boldsymbol{z}^q_{k+j+1|k,c} \in \mathcal{Z}^p_{j,c}, \qquad \forall c. \tag{5.46}$$

By using (5.12) and (5.44b), $\forall j$,

$$\begin{aligned}
\hat{\boldsymbol{z}}^{p+1}_{k+j+1|k+1,c} &= E^{p+1}_c \hat{\boldsymbol{x}}^{p+1}_{k+j+1|k+1} \\
&= E^{p+1}_c \boldsymbol{x}^{p+1}_{k+j+1|k} + E^{p+1}_c L^{p+1}_j \boldsymbol{w}^{p+1}_k \\
&= \boldsymbol{z}^{p+1}_{k+j+1|k,c} + E^{p+1}_c L^{p+1}_j \boldsymbol{w}^{p+1}_k.
\end{aligned}$$

Then, (5.46) becomes

$$\sum_{q=1}^{p} \boldsymbol{z}^q_{k+j+1|k+1,c} + \hat{\boldsymbol{z}}^{p+1}_{k+j+1|k+1,c} - E^{p+1}_c L^{p+1}_j \boldsymbol{w}^{p+1}_k + \sum_{q=p+2}^{n} \boldsymbol{z}^q_{k+j+1|k,c} \in \mathcal{Z}^p_{j,c}. \tag{5.47}$$

Using (5.28b),

$$\begin{aligned}
\sum_{q=1}^{p} \boldsymbol{z}^q_{k+j+1|k+1,c} + \hat{\boldsymbol{z}}^{p+1}_{k+j+1|k+1,c} + \sum_{q=p+2}^{n} \boldsymbol{z}^q_{k+j+1|k,c} &\in \mathcal{Z}^p_{j,c} \oplus E^{p+1}_c L^{p+1}_j \mathcal{W}^{p+1} \\
&\subseteq \mathcal{Z}^{p+1}_{j,c}.
\end{aligned} \tag{5.48}$$

Finally, we have

$$\hat{\boldsymbol{z}}^{p+1}_{k+j+1|k+1,c} + \tilde{\boldsymbol{z}}^{p+1}_{k+j+1|k+1,c} \in \mathcal{Z}^{p+1}_{j,c} \tag{5.49}$$

satisfying the coupling constraints for vehicle $p+1$.

5.C MILP Implementation of DRSBK Optimization

This appendix shows the detailed MILP implementation of DRSBK algorithm. The disturbance in this section is assumed to be infinity-norm bounded, i.e., $\mathcal{W}^p = \{\boldsymbol{w} \mid \boldsymbol{w} = G\boldsymbol{n},\ \|\boldsymbol{n}\|_\infty \leq w_{\max}\}$.

Constraint Tightening for Robustness

The constraint tightening in (5.17) and (5.28) are implemented using the following constraint contraction parameters [102]

$$\begin{aligned}
&\alpha_0 = 0, && \alpha_j = \alpha_{j-1} + \left\|[1\ 0\ 0\ 0]L_{j-1}^p B^p G\right\|_1 w_{\max}, \quad j \geq 1 \\
&\beta_0 = 0, && \beta_j = \beta_{j-1} + C\left\|[0\ 0\ 1\ 0]L_{j-1}^p B^p G\right\|_1 w_{\max}, \quad j \geq 1 \\
&\gamma_0 = 0, && \gamma_j = \gamma_{j-1} + C\left\|[1\ 0]P_j^p B^p G\right\|_1 w_{\max}, \quad j \geq 1
\end{aligned} \tag{5.50}$$

where α_j, β_j, and γ_j respectively represents the constraint contraction for position, velocity, and input for the j^{th} prediction step. The coefficient $C = 1$ when the constraint set (5.32) and (5.33) and the disturbance set are both two-norm bounded. However, $C = \sqrt{2}$ when performing the Pontryagin difference between a two-norm bounded set and the infinite-norm bounded disturbance set $\mathcal{W}^p$ in (5.30). This is because $\mathcal{W}^p$ has the maximum magnitude of the length $\sqrt{2}w_{\max}$ in the diagonal directions.

Output Constraint Set (5.11)

The obstacle avoidance constraints use binary variables. For each point $\boldsymbol{r}_{k+j|k}^p = [x_{k+j|k}^p, y_{k+j|k}^p]^T$ and each rectangular shaped obstacle defined by two corners $[x_{\text{low}}, y_{\text{low}}]^T$

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

$$\forall o,\ \forall j: \quad x^p_{k+j|k} \le x_{\text{low},o} - \alpha_j + M\, b^p_{\text{obst},jo1} \tag{5.51a}$$

$$y^p_{k+j|k} \le y_{\text{low},o} - \alpha_j + M\, b^p_{\text{obst},jo2} \tag{5.51b}$$

$$x^p_{k+j|k} \ge x_{\text{high},o} + \alpha_j - M\, b^p_{\text{obst},jo3} \tag{5.51c}$$

$$y^p_{k+j|k} \ge y_{\text{high},o} + \alpha_j - M\, b^p_{\text{obst},jo4} \tag{5.51d}$$

$$\sum_{i=1}^{4} b^p_{\text{obst},joi} \le 3 \tag{5.51e}$$

where M is a large number to relax the constraints in (5.51a)–(5.51d), and o denotes the index of the obstacle. The logical constraint (5.51e) requires at least one constraint in (5.51a)–(5.51d) be active. Note that the parameter α_j tightens the constraints by enlarging the obstacles.

The output constraint (5.11) also includes the bound on speed and inputs. Let vectors $\boldsymbol{r}, \boldsymbol{v}$, and $\boldsymbol{a}$ respectively represent position, velocity, and acceleration input in the inertia frame. A set of n_d linear constraints approximates the two-norm bounded constraints on the acceleration and velocity vectors, which in turn limits the maximum turning rate

$$\left[\cos\theta_m,\ \sin\theta_m\right] \boldsymbol{v}^p_{k+j|k} \le v_{\max} - \beta_j \tag{5.52a}$$

$$\left[\cos\theta_m,\ \sin\theta_m\right] \boldsymbol{a}^p_{k+j|k} \le a_{\max} - \gamma_j \tag{5.52b}$$

$$\theta_m = \frac{2\pi m}{n_d}, \quad \forall m = 1, \ldots, n_d.$$

The minimum speed constraint is non-convex and requires n_v binary variables to express in MILP

$$\left[\cos\theta_m,\ \sin\theta_m\right] \boldsymbol{v}^p_{k+j|k} \ge v_{\min} + \beta_j - 2v_{\max}\, b^p_{\text{vel},jm} \tag{5.53a}$$

$$\sum_{m=1}^{n_v} b^p_{\text{vel},jm} \le n_v - 1 \tag{5.53b}$$

One advantage of MILP is that the optimization can consider the entire range of

vehicle dynamics allowed by these constraints.

Invariance Constraints (5.14)

From the terminal states, the vehicle has an option to enter a left or right loiter circle. The centers of the left and right safety circles are

$$\boldsymbol{O}^p_{\mathrm{L}} = \boldsymbol{r}^p_{k+N|k} + R\left(\frac{\pi}{2}\right)\frac{\rho}{v_{\max} - \beta_{N-1}}\boldsymbol{v}^p_{k+N|k} \tag{5.54a}$$

$$\boldsymbol{O}^p_{\mathrm{R}} = \boldsymbol{r}^p_{k+N|k} + R\left(-\frac{\pi}{2}\right)\frac{\rho}{v_{\max} - \beta_{N-1}}\boldsymbol{v}^p_{k+N|k} \tag{5.54b}$$

where $R(\theta)$ is a rotation matrix of angle θ, and ρ is the radius of the turning circle given by

$$\rho = \frac{(v_{\max} - \beta_{N-1})^2}{a_{\max} - \gamma_{N-1}} \times \frac{v_{\max} - \beta_{N-1}}{v_{\min} + \beta_{N-1}}.$$

The second term accounts for the variability of the terminal speed $\|\boldsymbol{v}^p_{k+N|k}\|$. The binary variable b^p_{left} chooses either the left or right safety circle

$$\boldsymbol{O}^p_{\mathrm{L}} - 2(\rho + \alpha_{N-1})(1 - b^p_{\text{left}}) \leq \boldsymbol{O}^p \leq \boldsymbol{O}^p_{\mathrm{L}} - 2(\rho + \alpha_{N-1})(1 - b^p_{\text{left}}) \tag{5.55a}$$

$$\boldsymbol{O}^p_{\mathrm{R}} - 2(\rho + \alpha_{N-1})b^p_{\text{left}} \leq \boldsymbol{O}^p \leq \boldsymbol{O}^p_{\mathrm{R}} - 2(\rho + \alpha_{N-1})b^p_{\text{left}}. \tag{5.55b}$$

With the notation $\boldsymbol{O}^p = [x^p_{\text{center}},\ y^p_{\text{center}}]^{\mathrm{T}}$, the obstacle avoidance constraints of the safety circle are written as

$$\forall o: \quad x^p_{\text{center}} \leq x_{\text{low},o} - (\rho + \alpha_{N-1}) + M\, b^p_{\text{circ-obst},o1} \tag{5.56a}$$

$$y^p_{\text{center}} \leq y_{\text{low},o} - (\rho + \alpha_{N-1}) + M\, b^p_{\text{circ-obst},o2} \tag{5.56b}$$

$$x^p_{\text{center}} \geq x_{\text{high},o} + (\rho + \alpha_{N-1}) - M\, b^p_{\text{circ-obst},o3} \tag{5.56c}$$

$$y^p_{\text{center}} \geq y_{\text{high},o} + (\rho + \alpha_{N-1}) - M\, b^p_{\text{circ-obst},o4} \tag{5.56d}$$

$$\sum_{i=1}^{4} b^p_{\text{circ-obst},oi} \leq 3. \tag{5.56e}$$

Interconnected Constraints (5.13)

Over the planning horizon, the coupling constraints include vehicle avoidance constraints

$$x^p_{k+j|k} \leq x^q_{k+j|k} - d^{pq}_{\text{total}} + M\, b^{pq}_{\text{veh},j1} \tag{5.57a}$$

$$y^p_{k+j|k} \leq y^q_{k+j|k} - d^{pq}_{\text{total}} + M\, b^{pq}_{\text{veh},j2} \tag{5.57b}$$

$$x^p_{k+j|k} \geq x^q_{k+j|k} + d^{pq}_{\text{total}} - M\, b^{pq}_{\text{veh},j3} \tag{5.57c}$$

$$y^p_{k+j|k} \geq y^q_{k+j|k} + d^{pq}_{\text{total}} - M\, b^{pq}_{\text{veh},j4} \tag{5.57d}$$

$$\sum_{i=1}^{4} b^{pq}_{\text{veh},ji} \leq 3 \tag{5.57e}$$

where

$$d^{pq}_{\text{total}} = \begin{cases} 2d + 2\alpha_j, & q < p \\ 2d + \alpha_j + \alpha_{j+1}, & q > p. \end{cases}$$

Beyond the planning horizon, constraints on the safety circles ensure the vehicle avoidance

$$x^p_{\text{center}} \leq x^q_{\text{center}} - 2(\rho + d + \alpha_{N-1}) + M\, b^{pq}_{\text{circ},1} \tag{5.58a}$$

$$y^p_{\text{center}} \leq y^q_{\text{center}} - 2(\rho + d + \alpha_{N-1}) + M\, b^{pq}_{\text{circ},2} \tag{5.58b}$$

$$x^p_{\text{center}} \geq x^q_{\text{center}} + 2(\rho + d + \alpha_{N-1}) - M\, b^{pq}_{\text{circ},3} \tag{5.58c}$$

$$y^p_{\text{center}} \geq y^q_{\text{center}} + 2(\rho + d + \alpha_{N-1}) - M\, b^{pq}_{\text{circ},4} \tag{5.58d}$$

$$\sum_{i=1}^{4} b^{pq}_{\text{circ},i} \leq 3. \tag{5.58e}$$

Objective Function (5.20)

The objective function uses a binary variable $\boldsymbol{b}^p_{\text{vis}}$ to select one visible point $\boldsymbol{r}^p_{\text{vis}}$ from a list of cost points, from which the cost-to-go is known. Let $\boldsymbol{r}_{\text{cp},i}$ denote the i^{th} cost

point and $i = 1, \ldots, n_{\text{cp}}$ where n_{cp} is a number of cost points. Then,

$$\boldsymbol{r}^p_{\text{vis}} = \sum_{i=1}^{n_{\text{cp}}} b^p_{\text{vis},i}\ \boldsymbol{r}^p_{\text{cp},i} \tag{5.59a}$$

$$\sum_{i=1}^{n_{\text{cp}}} b^p_{\text{vis},i} = 1 \tag{5.59b}$$

$$\tilde{f}^p(\boldsymbol{r}^p_{\text{vis}}) = \sum_{i=1}^{n_{\text{cp}}} b^p_{\text{vis},i}\ \tilde{f}^p(\boldsymbol{r}^p_{\text{cp},i}) \tag{5.59c}$$

where the cost-to-go $\tilde{f}^p(\boldsymbol{r}^p_{\text{cp},i})$ from each cost point to the target of vehicle p is calculated prior to MILP and is constant in MILP. The objective value J^p to be minimized is a sum of two-norm distance from the terminal point $\boldsymbol{r}^p_{k+N|k}$ to the selected cost point $\boldsymbol{r}^p_{\text{vis}}$ and the cost-to-go from there

$$J^p \geq [\cos\theta_m,\ \sin\theta_m](\boldsymbol{r}^p_{k+N|k} - \boldsymbol{r}^p_{\text{vis}}) + \tilde{f}^p(\boldsymbol{r}^p_{\text{vis}}), \quad \forall m. \tag{5.60}$$

To ensure the visibility of the selected cost point $\boldsymbol{r}^p_{\text{vis}}$ from the terminal point $\boldsymbol{x}^p_{k+N|k}$, obstacle avoidance constraints are enforced on n_{int} interpolation points that are placed on the line connecting $\boldsymbol{r}^p_{\text{vis}}$ and $\boldsymbol{x}^p_{k+N|k}$

$$\mu_l x^p_{k+N|k} + (1-\mu_l)x^p_{\text{vis}} \leq x_{\text{low},o} - \alpha_{N-1} + M\, b^p_{\text{int},lo1} \tag{5.61a}$$

$$\mu_l y^p_{k+N|k} + (1-\mu_l)y^p_{\text{vis}} \leq y_{\text{low},o} - \alpha_{N-1} + M\, b^p_{\text{int},lo2} \tag{5.61b}$$

$$\mu_l x^p_{k+N|k} + (1-\mu_l)x^p_{\text{vis}} \geq x_{\text{high},o} + \alpha_{N-1} - M\, b^p_{\text{int},lo3} \tag{5.61c}$$

$$\mu_l y^p_{k+N|k} + (1-\mu_l)y^p_{\text{vis}} \geq y_{\text{high},o} + \alpha_{N-1} - M\, b^p_{\text{int},lo4} \tag{5.61d}$$

$$\sum_{i=1}^{4} b^p_{\text{int},loi} \leq 3 \tag{5.61e}$$

$$\mu_l = \frac{l}{n_{\text{int}}}, \qquad l = 1, \ldots, n_{\text{int}}.$$

In summary, the MILP implementation of subproblem $\mathsf{P}^p(\boldsymbol{x}^p_k)$ is to minimize J^p in (5.60) subject to (5.9)–(5.10), (5.51)–(5.61). The optimization variables are $\boldsymbol{r}^p_{k+j|k}$, $\boldsymbol{v}^p_{k+j|k}$, $\boldsymbol{a}^p_{k+j|k}$, $\boldsymbol{O}^p_{\text{L}}$, $\boldsymbol{O}^p_{\text{R}}$, $\boldsymbol{O}^p$, $\boldsymbol{r}^p_{\text{vis}}$, and all binary variables b^p's.

Chapter 6

Cooperative Decentralized Trajectory Optimization with Coupling Constraints

Motivated by recent research on cooperative UAVs, this chapter introduces a new decentralized trajectory optimization approach for systems with independent dynamics but coupled constraints. The primary objective is to improve the performance of the entire fleet by solving local optimization problems. The challenge here is how to coordinate the vehicles without reproducing the global optimization problem for each agent. To achieve cooperation, the approach exploits the sparse structure of active couplings that is inherent in the trajectory optimization. This enables each local optimization to use a low-order parameterization of the other agents states, thereby facilitating negotiation while keeping the problem size small. The key features of this approach include (a) no central negotiator is required; and (b) it maintains feasibility over the iterations, so the algorithm can be stopped at any time. Furthermore, the local optimizations are shown to monotonically decrease the overall cost. Simulation results are presented to compare the distributed, centralized, and other (non-cooperative) decentralized approaches in terms of both computation and performance.

6.1 Introduction

Much of the current research on decentralized trajectory optimization uses a setup where each vehicle optimizes only for its own control and communicates this intent information to its neighbors [62, 71, 110, 112–115], as represented by the DRSBK algorithm presented in Chapter 5. These decentralized algorithms typically lead to a Nash equilibrium [116] or a Pareto optimal surface [113, 117], which is not necessarily the globally optimal solution, because these so-called "communication-based approaches" [115] do not use the information about the cost functions of other subsystems and do not consider the overall performance.

A typical cooperative behavior is to sacrifice the individual objective if it benefits the overall team performance. The challenge in achieving such fleet level cooperation is how to address the global performance in the decentralized planning setup. For example, previous work by Inalhan [113] softens the constraints and achieves fleet level agreement by iteratively increasing the penalty on the constraint violation, but this iteration process could take a long time before it even reaches a feasible solution. Another iterative decentralized scheme has been recently proposed [115] for systems coupled through their dynamics to achieve a cooperative solution. A dual decomposition approach has been proposed for systems coupled through objectives [118]. This chapter focuses on problems with independent dynamics but with coupling constraints. The application examples include formation control of a fleet of UAVs, vehicle avoidance maneuvers, and multi-vehicle path planning under line-of-sight constraints. The proposed algorithm minimizes the global cost by solving local optimizations while satisfying all the constraints. The new approach in this chapter avoids the complexity of global optimization by using a reduced decision space for neighboring systems in each local optimization. In particular, it exploits the problem structure to parameterize the other vehicles' decisions using the active coupling constraints. This approach is suitable for trajectory optimization because it typically has only a few active couplings.

The chapter is organized as follows. First, Section 6.2 introduces the overall

problem and two straightforward approaches. In Section 6.3.1, a simple form of the proposed algorithm is presented first to highlight the implication of this approach. Section 6.3.2 presents the complete decentralized cooperative algorithm. Finally, Section 6.5 shows the simulation results and compares the algorithm with other available approaches in terms of performance and computation time.

6.2 Problem Statement

The problem of interest is a general optimization for multi-vehicle systems, with a particular emphasis on path planning. A fleet of n vehicles are assumed to have independent dynamics. In this chapter, the superscript or subscript i, j denote the vehicle index. Different types of constraints are imposed, but they can be divided into (a) local constraints, such as speed bounds, input saturation, and obstacle avoidance; and (b) coupling constraints such as vehicle avoidance, inter-vehicle communication range, and line-of-sight between vehicles.

The system dynamics are in discrete time, and an optimization is performed to obtain the optimal input for each vehicle over N steps into the future:

$$\forall i = 1, \ldots, n, \quad \forall k = 0, \ldots, N-1:$$

$$\boldsymbol{x}_{k+1}^i = f_i\left(\boldsymbol{x}_k^i, \boldsymbol{u}_k^i\right) \tag{6.1}$$

$$g_i\left(\boldsymbol{x}_k^i, \boldsymbol{u}_k^i\right) \leq 0 \tag{6.2}$$

$$A_i \boldsymbol{x}_k^i + A_j \boldsymbol{x}_k^j \leq b_{ij}, \qquad \forall j = i+1, \ldots, n \tag{6.3}$$

where $f_i\left(\boldsymbol{x}_k^i, \boldsymbol{u}_k^i\right)$ represents the nonlinear dynamics of vehicle i, and $g_i\left(\boldsymbol{x}_k^i, \boldsymbol{u}_k^i\right)$ represents all local constraints imposed on the states and the inputs of vehicle i. The pair-wise constraints (6.3) capture the coupling between all pairs of vehicles and are assumed to be a combination of linear constraints, which can express various types of polyhedral constraints including the 1-norm, the approximate 2-norm, and the ∞-norm bounds.

The objective function for the entire fleet is a sum of the individual costs, which

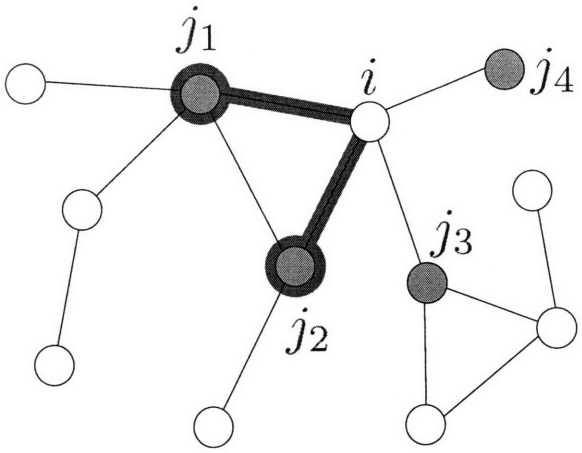


Figure 6-1: Node i's neighbor set $\mathcal{N}$ and active coupling neighbor set $\mathcal{A}$.

could be competing with each other

$$\min_{\boldsymbol{u}(\cdot)} \sum_{i=1}^{n} J_i \tag{6.4}$$

$$\text{s.t.} \quad J_i = \sum_{k=0}^{N-1} l_i\big(\boldsymbol{x}_k^i, \boldsymbol{u}_k^i\big) + F_i\big(\boldsymbol{x}_N^i\big), \quad \forall i \tag{6.5}$$

where $l_i\big(\boldsymbol{x}_k^i, \boldsymbol{u}_k^i\big)$ is a stage cost and $F_i\big(\boldsymbol{x}_N^i\big)$ is the terminal penalty.

6.2.1 Notation

In this chapter, we define the term *neighbor* of vehicle i as a set of vehicles that have any coupling constraint with vehicle i. In particular, if $\mathcal{N}(i)$ denotes this neighbor set for vehicle i, then there exists a coupling constraint between the two vehicles i and $j \in \mathcal{N}(i)$. Furthermore, let $\mathcal{A}(i)$ denote a set of vehicles that have active coupling constraints with vehicle i. Figure 6-1 shows an example of a graphical representation of a vehicle fleet. Each node represents a vehicle, and the arc connecting two nodes shows that there is a coupling constraint between the two vehicles. The shaded nodes in the figure are the neighbors of vehicle i, i.e., $j_1, \ldots, j_4 \in \mathcal{N}(i)$. In general, not all of the coupling constraints are active. In this example, active coupling neighbors of vehicle i are marked with thick lines, and $j_1, j_2 \in \mathcal{A}(i)$ but $j_3, j_4 \notin \mathcal{A}(i)$.

For notational simplicity, let z_i denote the decision variable of the i^{th} vehicle, i.e.,

$z_i \triangleq \left[\boldsymbol{u}_0^{i\,T}, \cdots, \boldsymbol{u}_{N-1}^{i\,T} \right]^T$. Then, with some abuse of notation, a compact form of the optimization (6.1)–(6.5) can be written as

$$\begin{aligned} \min_{z_1,\ldots,z_n} \quad & \sum_{i=1}^{n} J_i(z_i) \\ \text{subject to} \quad \forall i: \quad & g(z_i) \leq 0 \\ & h(z_i, z_j) \leq 0, \quad j \in \mathcal{N}(i) \end{aligned} \tag{6.6}$$

where $g(z_i)$ represents the local constraints for vehicle i, and $h(z_i, z_j)$ represents the coupling constraints between vehicles i and j.

6.2.2 Centralized Approach

The centralized approach directly solves the full (and potentially large) optimization given in (6.6). This approach produces the globally optimal solution; however, it scales poorly because the optimization becomes very complex for large fleets for most problem types (i.e., quadratic programming and mixed-integer linear programming) [102].

6.2.3 Decentralized Non-cooperative Approach

One decentralized approach is to decompose the centralized problem (6.6) into smaller subproblems, as discussed in Chapter 5. Figure 6-2 shows the procedural flow. Similar to Gauss-Seidel iteration [119], the approach sequentially solves the local subproblems and sends the solutions to other vehicles. In a subproblem, each vehicle i freezes the other vehicles' decision variables and solves for its own optimal input $z_i{}^*$. The local optimization for vehicle i can then be written as

$$\begin{aligned} \min_{z_i} \quad & J_i(z_i) \\ \text{subject to} \quad & g(z_i) \leq 0 \\ & h(z_i, \bar{z}_j) \leq 0, \quad j \in \mathcal{N}(i). \end{aligned} \tag{6.7}$$

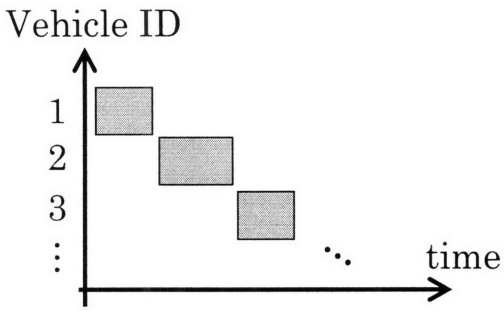


Figure 6-2: Decentralized sequential planning. Each gray region represents that the vehicle is computing during that time.

The decision variables of the other vehicles are assumed to be constant, which is denoted as $\bar{z}_j$. The solution $z_i{}^*$ to the subproblem (6.7) is sent to the other vehicles, and the optimization of the next vehicle $i+1$ starts after receiving $z_i{}^*$.

The main advantage of this approach is that the subproblem has a smaller decision space (approximately n times smaller than the centralized approach), with many fewer constraints, as shown in Chapter 5. As a result, the computation time is much smaller and the algorithm scales much better than the centralized approach. Furthermore, all the constraints are satisfied while cycling through the vehicles. However, since each vehicle does not account for the objectives of other vehicles, the resulting solution is *coordinated* but *non-cooperative*. This non-cooperative solution is called a Nash equilibrium, where no vehicle can improve its local cost by changing only its own decision, and has to be avoided in the cooperative approach. These benefits and limitations of this approach are clearly illustrated in the examples in Section 6.5.

6.3 Decentralized Cooperative Optimization

This section presents a new decentralized cooperative algorithm. We first describe the approach for a simple version of the algorithm with only two vehicles i and j, one coupling constraint, and no local constraints.

6.3.1 Simple Version

Similar to the decentralized non-cooperative approach, each vehicle solves a local optimization problem in a sequential way by freezing the other vehicle's decisions when the coupling constraints are inactive. The key difference is that when there is an active coupling constraint, the new approach recognizes that each vehicle should consider sacrifices to its local performance if it is possible to reap a larger benefit to the overall team performance.

More formally, when vehicle i solves the optimization after vehicle j, the coupling constraint $a_i z_i + a_j \bar{z}_j \leq b_{ij}$ is modified, if active, in the following way

$$a_i z_i + a_j \bar{z}_j \leq b_{ij} - \beta. \tag{6.8}$$

The parameter β tightens the constraint for vehicle i if $\beta > 0$, which could make the local performance worse. However, vehicle i can account for the potential benefit to the other vehicle j by adding an extra term to the objective function

$$\min_{z_i, \beta} J_i(z_i) - \lambda \beta \tag{6.9}$$

where $\lambda \neq 0$ is a Lagrange multiplier of the coupling constraint that is obtained from the previous solution of the optimization by vehicle j. This Lagrange multiplier represents the amount of improvement (if $\lambda > 0$) or loss (if $\lambda < 0$) the optimization of vehicle j can obtain given some change in the right hand side of the coupling constraint $a_i \bar{z}_i + a_j z_j \leq b_{ij}$ [120]. In the hierarchical setup, the Lagrange multiplier could be used to represent the "price" of each vehicle's solution that the centralized negotiator can use to obtain a coordinated solution [7]. Note that this approach is meaningful only when the coupling constraint is active and hence $\lambda \neq 0$.

The decision variables of vehicle i's optimization in (6.9) are its local decisions z_i and the negotiation parameter β for the other vehicle. The parameter β allows vehicle i to sacrifice its local cost if that leads to more benefit to the other vehicle j, which corresponds to a *cooperative behavior*. This is the same as minimizing the

global performance by solving the local decentralized problems.

6.3.2 Full Version

This section generalizes the idea introduced in Section 6.3.1 to cases with multiple coupling and local constraints. When there are multiple active coupling constraints, simply including the effect $\lambda\beta$ for each active coupling constraint could doubly count the benefit that the other vehicle can obtain. Also, the scope of the simple problem must be expanded because it is possible to have vehicle i tighten its coupling constraint, as in (6.8), to enlarge the operating region of vehicle j, but a local constraint of vehicle j prevents it from using that extra region and obtaining the expected benefit $\lambda\beta$.

The simple version used the negotiation parameter β in vehicle i's optimization as an implicit decision variable for vehicle j. In the full version, vehicle i makes an explicit decision for vehicle j, which is denoted by δz_j. Then, vehicle i's optimization is

$$\min_{z_i, \delta z_j} J_i(z_i) + J_j(\bar{z}_j + \delta z_j) \tag{6.10}$$

$$\text{s.t.} \quad g(z_i) \leq 0 \tag{6.11}$$

$$g(\bar{z}_j + \delta z_j) \leq 0 \tag{6.12}$$

$$A_i z_i + A_j(\bar{z}_j + \delta z_j) \leq b_{ij} \tag{6.13}$$

where δz_j is related to β through $A_j \delta z_j \Leftrightarrow \beta$. The modified local optimization in (6.10) appears similar to the one solved in the centralized approach, but the key point is that, for the problems of interest in this work, the decision space can be reduced significantly, as discussed below.

Sparse active coupling

The approach avoids reproducing the global optimization problem for each agent by exploiting the structure of active couplings that is typical of trajectory optimization.

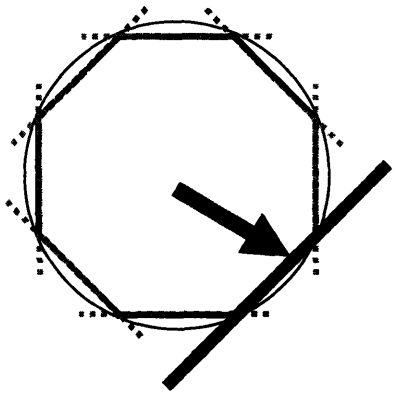


Figure 6-3: A two norm constraint approximated by a combination of linear constraints. The arrow shows a relative position vector. The thick line shows the only active coupling constraint.

For example, the vehicle avoidance constraints are imposed over all time steps of the plan, but they are typically active only at one or two time steps. Communication range limit constraints could be expressed as a nonlinear two-norm constraint on the relative position, but a combination of several linear constraints can approximate it. In such a case, as illustrated in Figure 6-3, only a few of the many existing constraints are active. The algorithm exploits this *sparse* structure of the active coupling constraints to reduce the size of the optimization problem.

Low-order parameterization

Without loss of generality, the upper rows of the coupling constraints (6.13) can be regarded as active and the lower rows as inactive.

$$\left[\begin{array}{c} A_j^{\text{active}} \\ \hline A_j^{\text{inactive}} \end{array}\right] \delta z_j = \left[\begin{array}{c} \beta^{\text{active}} \\ \hline * \end{array}\right]$$

We focus on changing β^{active} by δz_j, because a change in these active coupling constraints can lead to the direct change of the other vehicle's cost. This corresponds to focusing on the non-zero λ in the simple version. In order to address the change in β^{active}, a low-order parameterization of δz_j can be used because $\dim(\beta^{\text{active}}) \ll \dim(\delta z_j)$ in trajectory optimization.

Let m denote the row rank of A_j^{active}, which is also a number of elements in β^{active} that any δz_j can change independently. Therefore, a new variable $\alpha_j \in \mathbb{R}^m$ could replace δz_j, where in the trajectory optimization problems the dimension m of α_j is significantly smaller than the dimension of δz_j. Let $\check{A}$ denote a matrix composed of the m independent row vectors extracted from A_j^{active}. Then, δz_j is parameterized by α_j as

$$\delta z_j = \check{A}^T\left(\check{A}\check{A}^T\right)^{-1}\alpha_j \triangleq T_j\alpha_j. \tag{6.14}$$

The inverse in this equation exists because the product $\left(\check{A}\check{A}^T\right)$ is a matrix of full rank m, so the parameterization matrix T_j also exists.

With this new variable α_j, the local optimization can be rewritten as

$$\min_{\substack{z_i \\ \alpha_j,\, j\in\mathcal{A}(i)}} \left\{ J_i(z_i) + \sum_{j\in\mathcal{A}(i)} J_j(\bar{z}_j + T_j\alpha_j) \right\} \tag{6.15}$$

subject to

$$\begin{aligned}
g(z_i) &\leq 0 \\
g(\bar{z}_j + T_j\alpha_j) &\leq 0, \quad j \in \mathcal{A}(i) \\
h(z_i, \bar{z}_j) &\leq 0, \quad j \in \mathcal{N}(i),\ j \notin \mathcal{A}(i) \\
h(z_i, \bar{z}_j + T_j\alpha_j) &\leq 0, \quad j \in \mathcal{A}(i) \\
h(\bar{z}_k, \bar{z}_j + T_j\alpha_j) &\leq 0, \quad k \in \mathcal{N}(j), k \notin \mathcal{A}(i) \\
h(\bar{z}_{j_1} + T_{j_1}\alpha_{j_1}, \bar{z}_{j_2} + T_{j_2}\alpha_{j_2}) &\leq 0, \quad j_1, j_2 \in \mathcal{A}(i).
\end{aligned}$$

The parameterization is based on the active coupling constraints, but the optimization includes both the active and inactive constraints. The first two constraints are the local constraints for vehicle i and for its active coupling neighbors. The next four equations express different types of couplings shown in Figure 6-4. Type I is between vehicle i and its neighbors with no active couplings; Type II is between vehicle i and its neighbors with active couplings; Type III is between vehicle i's active coupling

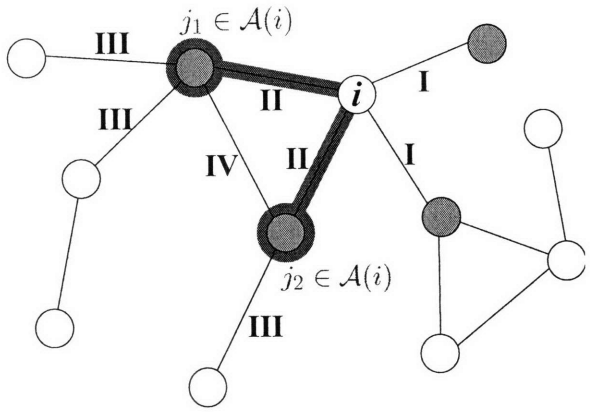


Figure 6-4: Different types of coupling constraints.

neighbors and their neighbors; Type IV is between vehicle i's two active coupling neighbors. Note that some constraints in $h(z_i, \bar{z}_j + T_j\alpha_j) \leq 0$ could be omitted if α_j has no impact on them because of the row rank deficiency of T_j. The key advantages of this algorithm are:

1. It does not freeze the other vehicle's plan, so that it can avoid Nash equilibrium;

2. It reduces the decision space of the other vehicles, so that the complexity of each local optimization remains low. For a problem of interest with a relatively large decision space (e.g., $\dim(z_i) = 20$, $n = 5$) and sparse active couplings (two active coupling neighbors, $\dim(\alpha) = 2$), the reduction of the decision space of each optimization would be a factor of ~ 4.

The algorithm iterates over the vehicles and the complete flow is summarized in Algorithm 6.1. Note that in the line 6 of Algorithm 6.1, the vehicle i makes a decision on itself and its neighbors. Then, line 7 ensures that all of the vehicles have the same values for the decision variables z_i and z_j. The local optimization (6.15) requires the knowledge of other vehicles' cost function, but this depends only on the target score, target location, vehicle states, etc., and is simple to communicate. The simulation results in Section 6.5 show that two iterations over the fleet produces a good performance that is comparable to the centralized approach.

Algorithm 6.1 Decentralized cooperation algorithm

1: Initialize the counter as 0
2: **for** each iteration **do**
3: **for** each vehicle $i = 1, \ldots, n$ **do**
4: Find the active coupling with this vehicle: $\mathcal{A}(i)$
5: Calculate the parameterization matrix $T_j,\ j \in \mathcal{A}(i)$
6: Solve local optimization (6.15) and obtain the solution $({z_i}^*, {\alpha_j}^*)$
7: Send the solution to other vehicles. Each vehicle updates the plan

$$z_i := {z_i}^*$$
$$z_j := z_j + T_j {\alpha_j}^*, \quad j \in \mathcal{A}(i)$$

8: **if** the counter is larger than the maximum number of iterations **then**
9: Terminate
10: **else**
11: Increment the counter by 1 and go to the next vehicle
12: **end if**
13: **end for**
14: **end for**

6.4 Algorithm Properties

This section discusses two key properties of this algorithm.

6.4.1 Feasibility Over the Iteration

First, we show that feasibility is maintained while cycling through the vehicles.

Theorem 6.1. *Assume the fleet initially satisfies all of the constraints*

$$\forall j: \quad g(z_j) \leq 0$$
$$h(z_j, z_k) \leq 0, \quad k \in \mathcal{N}(j).$$

Then, if the optimization of vehicle i is feasible, the optimization of the next vehicle $i+1$ is also feasible.

Proof. Let the superscript $(\cdot)^\circ$ represents the decision variable prior to vehicle i's

optimization, and the superscript $(\cdot)^*$ represents the solution optimized by i. Initially,

$$\begin{aligned} \forall j: \quad & g(z_j^\circ) \leq 0 \\ & h(z_j^\circ, z_k^\circ) \leq 0, \quad k \in \mathcal{N}(j). \end{aligned}$$

The feasible solution of vehicle i's optimization that is obtained in step 6 of Algorithm 6.1 satisfies the following constraints.

$$\begin{aligned} g(z_i{}^*) &\leq 0 \\ g(z_j^\circ + \delta z_j^*) &\leq 0, \quad j \in \mathcal{A}(i) \\ h(z_i^*, z_j^\circ) &\leq 0, \quad j \in \mathcal{N}(i),\ j \notin \mathcal{A}(i) \\ h(z_i^*, z_j^\circ + \delta z_j^*) &\leq 0, \quad j \in \mathcal{A}(i) \\ h(z_k^\circ, z_j^\circ + \delta z_j^*) &\leq 0, \quad k \in \mathcal{N}(j), k \notin \mathcal{A}(i) \\ h(z_{j_1}^\circ + \delta z_{j_1}^*, z_{j_2}^\circ + \delta z_{j_2}^*) &\leq 0, \quad j_1, j_2 \in \mathcal{A}(i) \end{aligned}$$

After communicating the solutions and updating them in step 7 of Algorithm 6.1, the variables satisfy all the constraints

$$\forall i: \quad g(\bar{z}_i) \leq 0 \tag{6.16a}$$

$$h(\bar{z}_i, \bar{z}_j) \leq 0, \quad \forall j \in \mathcal{N}(i) \tag{6.16b}$$

where

$$\begin{aligned} \bar{z}_i &= z_i^* \\ \bar{z}_j &= z_j^\circ + \delta z_j^*, \quad j \in \mathcal{A}(i) \\ \bar{z}_j &= z_j^\circ, \qquad\quad j \notin \mathcal{A}(i). \end{aligned}$$

Given these results, the constraints for the problem for the next vehicle in the opti-

mization sequence $(i+1)$ are:

$$g(z_{i+1}) \leq 0 \tag{6.17a}$$

$$g(\bar{z}_j + \delta z_j) \leq 0, \quad j \in \mathcal{A}(i+1) \tag{6.17b}$$

$$h(z_{i+1}, \bar{z}_j) \leq 0, \quad j \in \mathcal{N}(i+1),\ j \notin \mathcal{A}(i+1) \tag{6.17c}$$

$$h(z_{i+1}, \bar{z}_j + \delta z_j) \leq 0, \quad j \in \mathcal{A}(i+1) \tag{6.17d}$$

$$h(\bar{z}_k, \bar{z}_j + \delta z_j) \leq 0, \quad k \in \mathcal{N}(j), k \notin \mathcal{A}(i+1) \tag{6.17e}$$

$$h(\bar{z}_{j_1} + \delta z_{j_1}, \bar{z}_{j_2} + \delta z_{j_2}) \leq 0, \quad j_1, j_2 \in \mathcal{A}(i+1). \tag{6.17f}$$

By comparing (6.16) and (6.17), it can be shown that reusing the solution from vehicle i's optimization

$$\begin{aligned} z_{i+1} &= \bar{z}_{i+1}, \\ \delta z_j &= 0, \quad \forall j \in \mathcal{A}(i+1) \end{aligned}$$

provides a feasible solution to the optimization of $i+1$. □

Note that because the feasibility of the entire fleet is maintained over the iteration, the algorithm could be terminated at any time. Previous work [62, 110] also maintains this feasibility property, but not the following property about the performance.

6.4.2 Monotonically Decreasing Global Cost

This subsection shows that the global objective value is monotonically decreasing along the iteration.

Theorem 6.2. *The global cost function defined by*

$$J(\boldsymbol{z}) = \sum_{i=1}^{n} J_i(z_i)$$

is monotonically decreasing in Algorithm 6.1, while cycling through the vehicles (line 3) and over the iteration (line 2).

Proof. The local optimization of vehicle i results in

$$J_i(z_i^\circ) + \sum_{j \in \mathcal{A}(i)} J_j(z_j^\circ) \geq J_i(z_i^*) + \sum_{j \in \mathcal{A}(i)} J_j(z_j^\circ + \delta z_j^*)$$

Let $J(\boldsymbol{z}^\circ)$ denote the global cost prior to the optimization by vehicle i and $J(\bar{\boldsymbol{z}})$ denote the global cost prior to the optimization by the next vehicle $i+1$. Then,

$$\begin{aligned} J(\boldsymbol{z}^\circ) &= J_i(z_i^\circ) + \sum_{j \in \mathcal{A}(i)} J_j(z_j^\circ) + \sum_{j \notin \mathcal{A}(i)} J_j(z_j^\circ) \\ &\geq J_i(z_i^*) + \sum_{j \in \mathcal{A}(i)} J_j(z_j^\circ + \delta z_j^*) + \sum_{j \notin \mathcal{A}(i)} J_j(z_j^\circ) \\ &= J_i(\bar{z}_i) + \sum_{j \in \mathcal{A}(i)} J_j(\bar{z}_j) + \sum_{j \notin \mathcal{A}(i)} J_j(\bar{z}_j) \quad = J(\bar{\boldsymbol{z}}) \end{aligned}$$

Therefore, each local optimization decreases the global cost. Because this algorithm maintains feasibility, it monotonically decreases the global cost while cycling through the vehicles and over the iteration. □

6.5 Simulation Results

6.5.1 Simulation Setup

In this simulation, all n vehicles are assumed to have the same linear dynamics which are described by a simple double integrator model: $\forall i = 1, \ldots, n$

$$\begin{aligned} \boldsymbol{x}_{k+1}^i &= \begin{bmatrix} O & I \\ O & O \end{bmatrix} \boldsymbol{x}_k^i + \begin{bmatrix} 0.5I \\ I \end{bmatrix} \boldsymbol{u}_k^i \\ \boldsymbol{x}_k^i &= \begin{bmatrix} \boldsymbol{r}_k^{i\,T} & \boldsymbol{v}_k^{i\,T} \end{bmatrix}^T. \end{aligned}$$

Constraints are imposed on the position, the speed, and the control input of each vehicle at each time step $k = 0, \ldots, N$

$$\left\|\boldsymbol{r}_k^i\right\|_2 \leq 1, \quad \left\|\boldsymbol{v}_k^i\right\|_2 \leq 0.35, \quad \left\|\boldsymbol{u}_k^i\right\|_2 \leq 0.18.$$

The systems are coupled with two neighbors through the following position constraints.

$$\begin{aligned} \left\|\boldsymbol{r}_k^i - \boldsymbol{r}_k^{i+1}\right\|_2 &\leq 0.8, \qquad i = 1, \ldots, n-1 \\ \left\|\boldsymbol{r}_k^n - \boldsymbol{r}_k^1\right\|_2 &\leq 0.8 \end{aligned}$$

These two-norm constraints are expressed as a combination of linear constraints. The cost direction for the i^{th} vehicle is

$$c_i = \left[\cos\left(\tfrac{i-1}{n}\right) \quad \sin\left(\tfrac{i-1}{n}\right) \right]^T.$$

The overall cost function to minimize is quadratic

$$\sum_{i=1}^{n} \sum_{k=0}^{N-1} \left\{ {\boldsymbol{x}_k^i}^T R_1 \boldsymbol{x}_k^i + {\boldsymbol{u}_k^i}^T R_2 \boldsymbol{u}_k^i \right\} + {c_i}^T \boldsymbol{r}_N^i + {\boldsymbol{r}_N^i}^T H \boldsymbol{r}_N^i$$

where the weights on the states R_1 and inputs R_2 in the stage cost are chosen to be much smaller than the weight H on the terminal position. Both the centralized and the local optimization are written as quadratic programming, and CPLEX 9.1 is used as a solver.

6.5.2 Simple Two Vehicle Case

The first example involves two vehicles i and j that can move on a two dimensional plane. The terminal position of the vehicle i has its local minimum at coordinates (0, 0.7), i.e.,

$$\begin{bmatrix} 0 \\ 0.7 \end{bmatrix} = \arg\min_{\boldsymbol{r}_N^i} \left\{ {c_i}^T \boldsymbol{r}_N^i + {\boldsymbol{r}_N^i}^T H \boldsymbol{r}_N^i \right\},$$

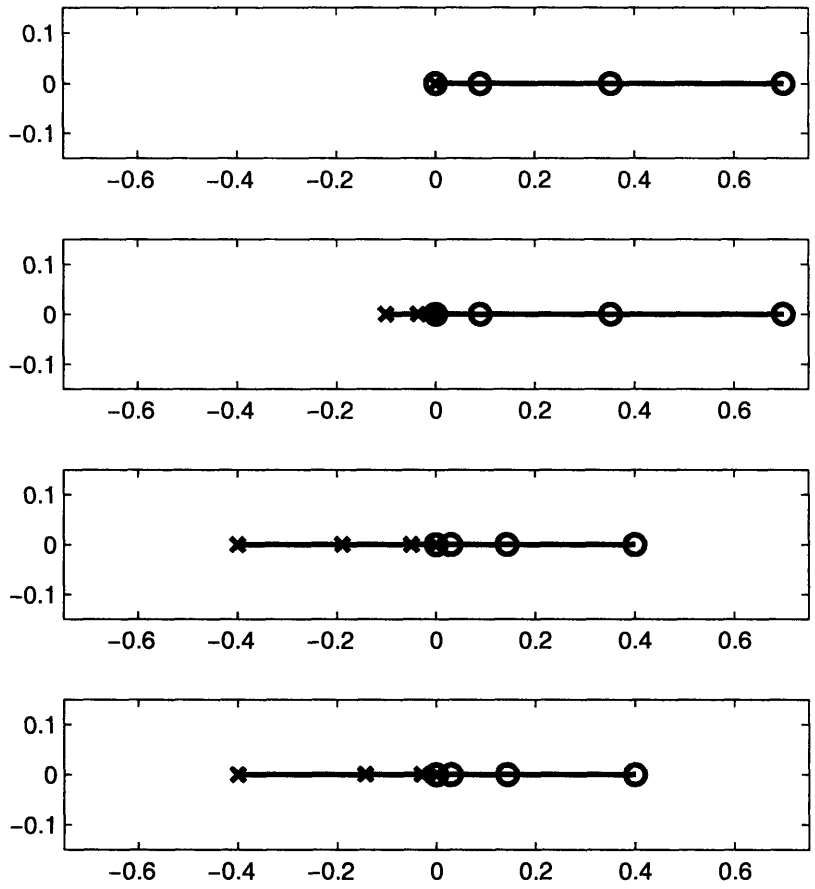


Figure 6-5: The evolution of plans over the iteration for the simple two vehicle example.

and that of the vehicle j is at $(0, -0.7)$. Because the two vehicles must satisfy the separation constraint of 0.8, their separate objectives are conflicting. The planning horizon is three steps for both vehicles.

Figure 6-5 shows the evolution of plans over two iterations. The plans of vehicle i are marked with $\circ$, and those of vehicle j are marked with $\times$. Originally, both vehicles are at the origin. First, vehicle i solves its local optimization. Because no coupling constraints are active at this point, the plan reaches the local minimum (0, 0.7). Vehicle j then solves its optimization, but given the separation constraint, this vehicle can only plan to move to $(0, -0.1)$, as shown in the second part of the figure.

The vehicle i solves the next optimization, but since a coupling constraint has become active, it uses a parameterized decision for j with a variable α_j of dimension $m = 1$. The bottom figure shows the plans after two iterations. The final plans are

the same as the globally optimal centralized solution.

If the decentralized non-cooperative algorithm in (6.7) were used, it would produce a Pareto optimal solution shown in the second figure of Figure 6-5, which is clearly not the globally optimal solution shown in the bottom. Note that if the vehicle j plans first followed by vehicle i, the non-cooperative algorithm results in a symmetric Pareto optimal solution, which again is not the globally optimal solution. This example clearly shows the performance improvement over the decentralized non-cooperative approach.

6.5.3 Five Vehicle Case

Figure 6-6 shows a more complex case with five vehicles. In this example, the local minimum for each vehicle is located on a unit circle centered on the origin. The planning horizon is three steps for all vehicles, and the planning order was $1 \rightarrow 3 \rightarrow 5 \rightarrow 2 \rightarrow 4$ to highlight the effect of the planning order on the performance.

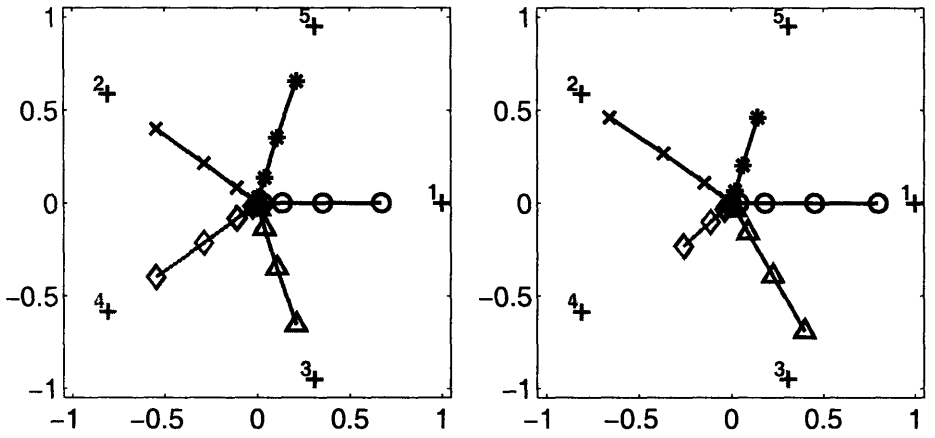
Two other algorithms described in Section 6.2 are used as benchmarks. These are: 1) the centralized approach (6.6) that provides the globally optimal solution, and 2) the decentralized non-cooperative approach (6.7) that produces a locally optimized solution.

As shown in Figure 6-6(b), the decentralized non-cooperative approach produced a suboptimal solution, because the vehicles that plan earlier are less constrained and have more region to operate than the vehicles that plan later in the cycle. The decentralized cooperation algorithm produced the trajectories shown in Figure 6-6(c) whose shape are very similar to the centralized solution shown in Figure 6-6(a).

6.5.4 Performance and Computation

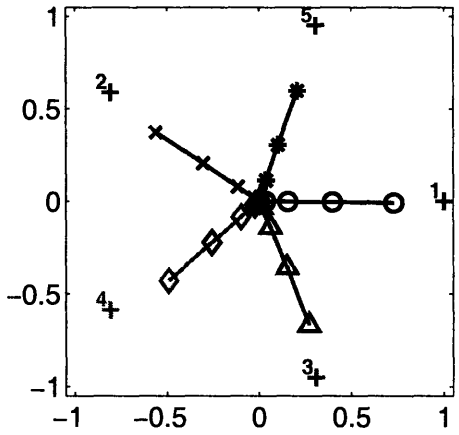
Figure 6-7 compares the global objective value and the cumulative computation time of three algorithms for the five vehicle example. Different lengths of the planning horizon $N = 4, 6, 8$ were considered to investigate the scalability of the algorithms.

The solutions of the decentralized non-cooperative approach are marked with $*$.



(a) Centralized solution

(b) Non-cooperative decentralized solution



(c) Cooperative decentralized solution

Figure 6-6: Final plans for five vehicles

Although the computation time is small, the cost is fairly high. The centralized (and hence globally optimal) solutions are marked with ∘. The lines with × show the evolution of the global cost of the decentralized cooperation algorithm. The plot starts from the end of the first iteration when every vehicle has its solution and continues to the end of the second iteration. This proposed algorithm has objective values comparable to those of the centralized solution but scales better than the centralized solution when the problem size increases.

Figure 6-8 shows cases with more vehicles (n = 5, 7, 10, 15). The decentralized non-cooperative approach has much higher cost and is out of the range of the plot. For the centralized and the proposed approach, the differences in the computation

time scale up significantly for larger fleets. Note that in all the plots of Figures 6-7 to 6-8, the lines of the proposed approach are monotonically decreasing, which validates the result in Section 6.4.2 by simulation.

6.6 Summary

This chapter presented a decentralized cooperation algorithm for systems coupled through the constraints. By exploiting the sparse structure of the active coupling constraints of trajectory optimization, the algorithm uses low-order parameterization of the decisions of neighboring vehicles. Simulation results showed that the proposed algorithm scales much better than the centralized approach and the performance is much better than that of the non-cooperative approach. Over the iteration, it is shown to maintain feasibility and monotonically decrease the global cost.

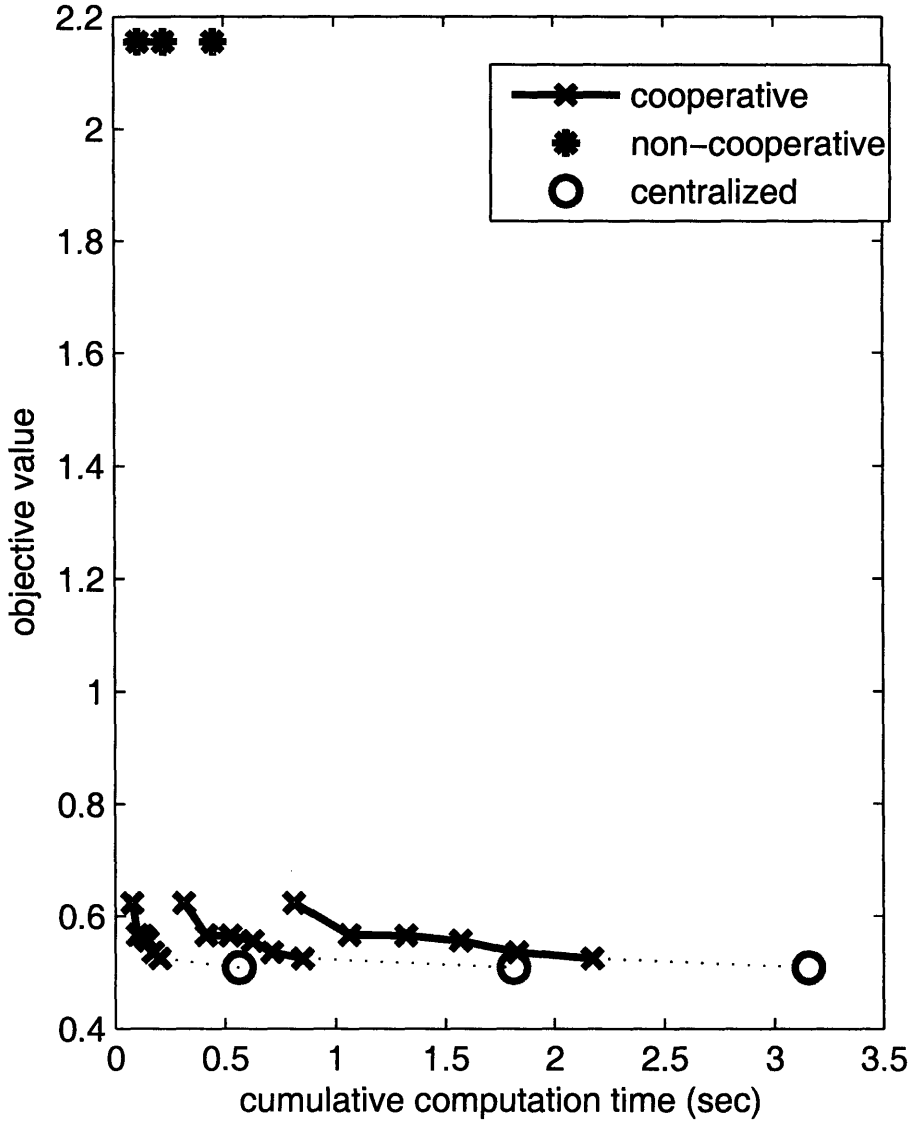


Figure 6-7: Comparison of three algorithms in terms of performance and computation.

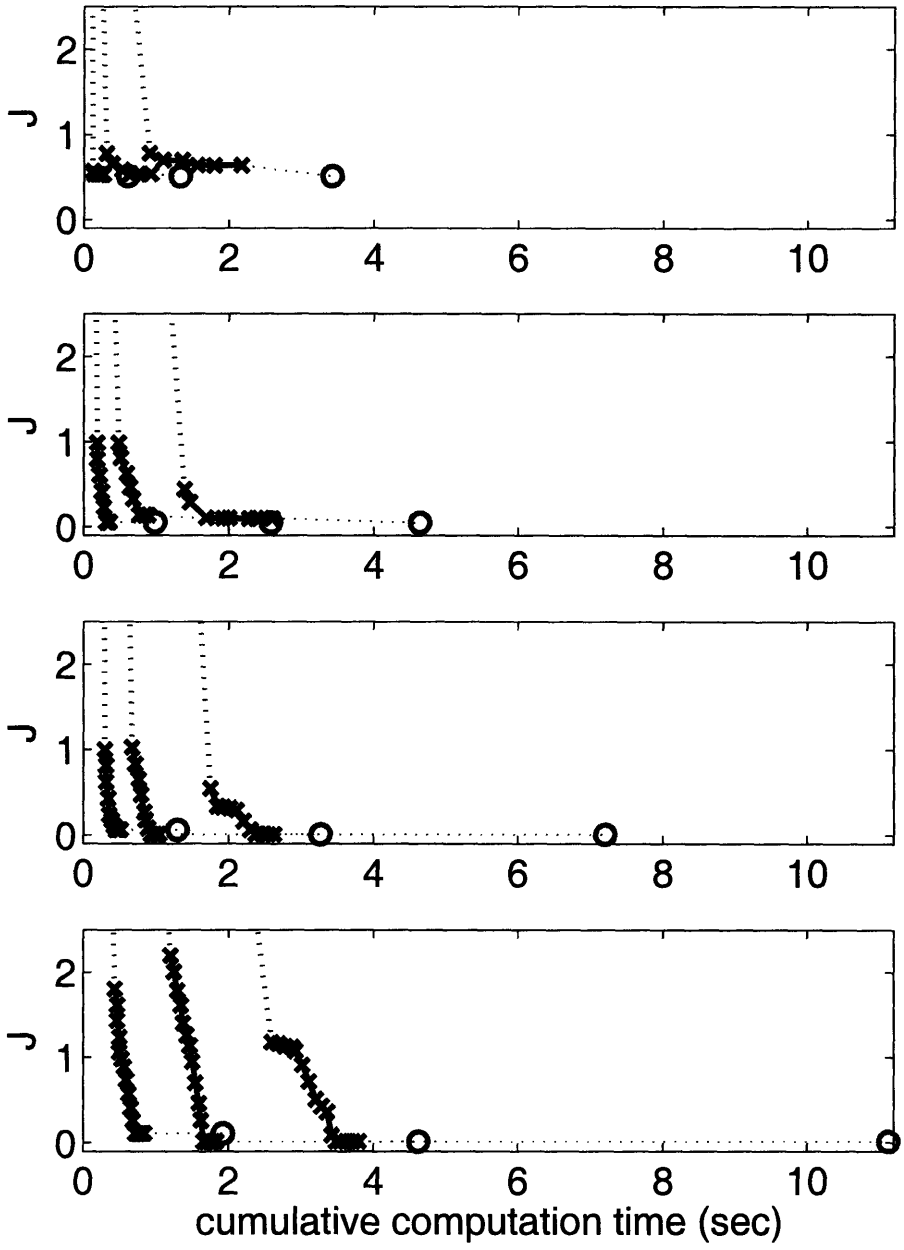


Figure 6-8: Trade-off between the performance and the computation time. From the top to the bottom, the number of vehicles are 5, 7, 10, and 15.

Chapter 7

Cooperative Decentralized Robust Trajectory Optimization using Receding Horizon MILP

The work in this chapter combines the decentralized cooperation technique presented in Chapter 6 with the distributed robust MPC path planner presented in Chapter 5 and then investigates how it extends to complex path planning problems using MILP. The overall goal is to develop a decentralized approach that solves small subproblems but minimizes a fleet-level objective. In this new algorithm, vehicles solve their subproblems in sequence, while generating feasible modifications to the prediction of other vehicles' plans. In order to avoid reproducing the global optimization, the decisions of other vehicles are parameterized using a much smaller number of variables than in the centralized formulation. The reduced number of variables is sufficient to improve the cooperation between vehicles without significantly increasing the computational effort involved. The resulting algorithm is shown to be robustly feasible under the action of unknown but bounded disturbances. Furthermore, the fleet objective is proven to monotonically decrease while cycling through the vehicles in the fleet and over the time. As an example of the cooperative behavior, the results from simulations and a hardware experiment demonstrate that the proposed algorithm can improve the fleet objective by temporarily having one vehicle sacrifice its individual

objective.

7.1 Introduction

In the multi-vehicle path planning problem, the path only needs to be designed locally around the vehicle, and several distributed control architectures have been proposed. Much of the current research on the distributed path planning assumes that each vehicle solves a local problem and communicates this intent information to its neighbors [62, 110, 112, 113]. In particular, DRSBK algorithm presented in Chapter 5 generates a local plan over a short horizon while guaranteeing the robust feasibility of the entire fleet under the action of bounded external disturbances [71]. This algorithm uses distributed robust MPC to predict and account for the behavior of other vehicles. However, because each vehicle only optimizes for its own control inputs and freezes the decisions of other vehicles, the resulting trajectories can be locally optimal but globally suboptimal (called the *non-cooperative solution*) [115].

This chapter extends the DRSBK algorithm to enable *cooperation* amongst the vehicles in the fleet. The approach builds on the decentralized cooperation technique presented in Chapter 6 and extends it to the more complex predictive control framework for path planning with coupling and obstacle avoidance constraints. Each vehicle solves its subproblem in sequence to optimize its own control input, as in DRSBK, but the subproblems also explicitly include the ability to modify the decisions of other vehicles to improve the global cost. The challenge here is to avoid reproducing the global optimization for each vehicle. The proposed approach uses a low-order parameterization of other vehicles to reduce the decision space while retaining the freedom to alter the key decisions of other vehicles. The DRSBK algorithm uses MILP in which binary variables represent the non-convex constraints and logical decisions in the trajectory optimization. The new algorithm presented in this chapter fixes most binary variables for other vehicles in a way that does not overly restrict the decision perturbation. This enables a negotiation with other vehicles through solving subproblems, with only a small increase in the computational complexity of

the subproblem. Another advantage of this approach is that this negotiation does not require iteration, and the algorithm is well-suited for real-time applications.

The chapter starts with a problem setup in Section 7.2. Section 7.3 presents the cooperative form of DRSBK algorithm. Section 7.4 proves that the new algorithm retains the robust feasibility and that each solution of the subproblem monotonically decreases the global cost. Finally, Section 7.5 shows simulation results, followed by the hardware results in Section 7.6.

7.2 Problem Statement

Notation: In this chapter, the index or superscript p, q denotes the vehicle index, index k denotes the current time step, and index j denotes the prediction step. There are total of n vehicles. Unless otherwise noted, $\forall p$ implies $\forall p = 1, \ldots, n$, and $\forall q$ implies $\forall q = 1, \ldots, n$ but $q \neq p$. The neighbor set $\mathcal{N}_k^p$ is a set of vehicles that have coupling constraints with vehicle p. It also determines the order that the vehicles solve their subproblems.

The vehicle dynamics are described by LTI model

$$\boldsymbol{x}_{k+1}^p = A\boldsymbol{x}_k^p + B\boldsymbol{u}_k^p + \boldsymbol{w}_k^p, \quad \forall p, \forall k \tag{7.1}$$

where $\boldsymbol{x}_k^p$ is the state vector of size n_x, $\boldsymbol{u}_k^p$ is the input vector of size n_u, and $\boldsymbol{w}_k^p$ is the disturbance vector for the p^{th} vehicle. The disturbances $\boldsymbol{w}_k^p$ are unknown but lie in known bounded sets $\boldsymbol{w}_k^p \in \mathcal{W}^p$. The environment has obstacles to be avoided, and the vehicles have flight envelope limitations. The general output sets $\mathcal{Y}^p$ capture these local constraints of each vehicle

$$\boldsymbol{y}_k^p = C\boldsymbol{x}_k^p + D\boldsymbol{u}_k^p \in \mathcal{Y}^p, \quad \forall p, \forall k. \tag{7.2}$$

The coupling between vehicles is captured by a further set of constraints applied to

the sum of outputs from each vehicle

$$\boldsymbol{z}_k^p = E^p \boldsymbol{x}_k^p, \quad \forall p \tag{7.3a}$$

$$\sum_{p=1}^{n} \boldsymbol{z}_k^p \in \mathcal{Z}. \tag{7.3b}$$

For pair-wise collision avoidance constraints, for a given row of the output matrix E, there are only two matrices E^p and E^q that have nonzero entries. The set $\mathcal{Z}$ is non-convex in this case.

Although the objective function can be an arbitrary function of all vehicles' decisions, this chapter assumes that the goal of the trajectory optimization is to minimize the mission completion time of the fleet. The summation of the individual cost is also included with a small penalty ϵ, so that the vehicles that complete the mission before the last vehicle also minimize the individual completion time.

$$\min J$$

$$J = \max_p J^p(\boldsymbol{x}^p, \boldsymbol{u}^p) + \epsilon \sum_{p=1}^{n} J^p(\boldsymbol{x}^p, \boldsymbol{u}^p) \tag{7.4}$$

7.3 Cooperative DRSBK Algorithm

This section presents a cooperative form of DRSBK algorithm using an approach that is similar to Chapter 6. The DRSBK algorithm described in Section 5.4 guarantees the robust constraint satisfaction under the action of disturbances. However, because the objective function (5.24) does not consider the effect from/on the other vehicles as in (7.4), the resulting solution could be a Nash equilibrium, where the solution is individually optimal but globally suboptimal. One intuitive approach to resolve this issue is to include all the decision variables of other vehicles in each subproblem. However, this will reproduce the global optimization for each vehicle and is clearly not scalable. The proposed approach explores the sparse structure of the coupling constraints of trajectory optimization and uses the low-order parameterization of the other vehicles to reduce the dimension of the decision space. The overall optimization

is implemented using MILP.

To simplify the presentation, some notation are introduced to represent the constraints of the DRSBK algorithms. Let $\mathsf{C}^p(\boldsymbol{x}_k^p, U_k^p)$ denote a set of constraints (5.9)–(5.16), (5.20) for vehicle p. The first argument $\boldsymbol{x}_k^p$ is the initial condition used in the right hand side of (5.9), and the second argument U_k^p denotes the control inputs of the predictive controller, i.e., $U_k^p = \left[{\boldsymbol{u}_{k|k}^p}^{\mathrm{T}}, \ldots, {\boldsymbol{u}_{k+N-1|k}^p}^{\mathrm{T}}\right]^{\mathrm{T}}$. From the optimal solution at time k, the first control input $\boldsymbol{u}_{k|k}^p$ for each vehicle is applied to the real system (7.1). At the next time $k+1$, the states of each vehicle $\boldsymbol{x}_{k+1}^p$ is measured, and the optimization is repeated.

7.3.1 Subproblem

Whereas DRSBK freezes all the decisions of other vehicles, the cooperative DRSBK (CDRSBK) updates the candidate solution of other vehicles by designing a feasible perturbation to other vehicles' decisions. Each vehicle sequentially solves its own subproblem as well as slightly modified subproblems of other vehicles. To simplify the presentation, this chapter assumes the planning order is $1, \ldots, n$.

Let $\tilde{\mathsf{C}}^p(\boldsymbol{x}_k^p, U_k^p)$ denote a set of constraints that is the same as $\mathsf{C}^p(\boldsymbol{x}_k^p, U_k^p)$ except that the tightened constraint equations (5.11) and (5.13) are replaced by the following

$$\forall j: \qquad \boldsymbol{y}_{k+j|k}^p \in \mathcal{Y}_{j+1}^p \tag{7.5}$$

$$\boldsymbol{z}_{k+j|k}^p + \tilde{\boldsymbol{z}}_{k+j|k}^p \in \mathcal{Z}_{j+1}^p \tag{7.6}$$

with $\mathcal{Y}_N^p \triangleq \mathcal{Y}_{N-1}^p$ and $\mathcal{Z}_N^p \triangleq \mathcal{Z}_{N-1}^p$, which are consistent with the tightening equations (5.17) and (5.28) under the nilpotency assumption $L_{N-1}^p = 0$. Then, the vehicle p solves the following optimization P_k^p:

$$\min_{\substack{\delta U_k^1, \ldots, \delta U_k^{p-1}, \\ U_k^p, \delta U_k^{p+1}, \ldots, \delta U_k^n}} \quad J$$

s.t. (7.4),

$$F\Big(\mathsf{C}^1(\boldsymbol{x}_k^1, \bar{U}_k^1 + \delta U_k^1), \ldots, \mathsf{C}^{p-1}(\boldsymbol{x}_k^{p-1}, \bar{U}_k^{p-1} + \delta U_k^{p-1}),\ \mathsf{C}^p(\boldsymbol{x}_k^p, U_k^p),$$
$$\tilde{\mathsf{C}}^{p+1}(\boldsymbol{x}_{k|k-1}^{p+1}, \bar{U}_k^{p+1} + \delta U_k^{p+1}), \ldots, \tilde{\mathsf{C}}^n(\boldsymbol{x}_{k|k-1}^n, \bar{U}_k^n + \delta U_k^n)\Big) \leq 0 \tag{7.7}$$

The function $F(\cdot)$ represents the constraints imposed on all vehicles. In this optimization, vehicle p designs its own control U_k^p as well as perturbation to other vehicles' control $\delta U_k^1, \ldots, \delta U_k^{p-1}$, $\delta U_k^{p+1}, \ldots, \delta U_k^n$. The notation $\bar{(\cdot)}^q$ denotes the variables that are received from other vehicles and are constant in vehicle p's subproblem, so that the control inputs for vehicle q are

$$U_k^q = \bar{U}_k^q + \delta U_k^q. \tag{7.8}$$

As shown in (7.7), if a vehicle q has already planned at time k, i.e., $q < p$, then the vehicle p uses $\mathsf{C}^q(\boldsymbol{x}_k^q, \bar{U}_k^q + \delta U_k^q)$ as the constraints for q. Otherwise, the constraints for q are $\tilde{\mathsf{C}}^q(\boldsymbol{x}_{k|k-1}^q, \bar{U}_k^q + \delta U_k^q)$, where the initial states are the states predicted in the previous plan. The objective is to minimize the fleet cost (7.4). After solving the optimization, the solution $(\cdot)^*$ updates the control as

$$\bar{U}_k^p := U_k^{p*}$$
$$\bar{U}_k^q := \bar{U}_k^q + \delta U_k^{q*}, \quad \forall q \neq p$$

which are sent to the next vehicle $p+1$. The formulation (7.7) may look similar to the centralized global optimization, but the next subsection discusses how to reduce the decision space.

7.3.2 Reduced-order Decision Space

Binary variables in MILP encode logical constraints or non-convex constraints but are the major source of the computational complexity in the MILP solution process. With the goal of obtaining small perturbations for other vehicles' decision, the CDRSBK algorithm freezes all of the binary variables of other vehicles, with one exception for

a loitering circle, as discussed later. Continuous decision variables are parameterized using the technique discussed in Chapter 6. A new decision variable ξ^q is introduced for the parameterized continuous decision of vehicle q ($\neq p$), so that

$$\delta U_k^q = T^q \xi^q$$

where T^q is a parameterization matrix. The dimension of ξ^q is much smaller than that of the original control input variables δU_k^q, so the row size of T^q is larger than the column size. In the examples considered in Sections 7.5 and 7.6, the row size is 2–6 times larger than the column size.

The performance of DRSBK algorithm depends strongly on the location of the invariant safety set that the terminal states $\boldsymbol{x}_{k+N|k}^q$ lies in. Thus, the terminal states $\boldsymbol{x}_{k+N|k}^q$ are selected as variables that are added to the subproblem of vehicle p. Since

$$\begin{aligned}\boldsymbol{x}_{k+N|k}^q &= \boldsymbol{x}_{k|k}^q + [A^{N-1}B, \cdots, AB, B] \begin{bmatrix} \boldsymbol{u}_{k|k}^q \\ \vdots \\ \boldsymbol{u}_{k+N-1|k}^q \end{bmatrix} \\ &\triangleq \boldsymbol{x}_{k|k}^q + \check{A} U_k^q,\end{aligned}$$

the perturbation in the control inputs are parameterized as

$$\delta U_k^q = \check{A}^{\mathrm{T}} (\check{A}\check{A}^{\mathrm{T}})^{-1} \delta \boldsymbol{x}_{k+N|k}^q \triangleq T^q \xi^q. \tag{7.9}$$

This parameterization matrix T^q always exists when the system is controllable and hence the matrix $\check{A}$ has full row rank. The parameterization (7.9) reduces the dimension of the decision space from δU_k^q, which is Nn_u, to $\delta \boldsymbol{x}_{k+N|k}^q$, which is n_x. Note that if vehicles p and q are far apart and do not interact with each other, the perturbation δU_k^q can be set to zero in p's optimization.

If the terminal invariant constraint requires the vehicle to stop at the terminal step of the plan, as in the hovering constraint for rotorcraft, the parameterization matrix must be formed differently. Let $C_{\mathrm{pos}} \triangleq [I,\ O]$ and $C_{\mathrm{vel}} \triangleq [O,\ I]$ for a double

integrator system, where

$$\delta \boldsymbol{x}^q_{k+N|k} = \begin{bmatrix} \delta \boldsymbol{r}^q_{k+N|k} \\ \delta \boldsymbol{v}^q_{k+N|k} \end{bmatrix}$$

In order to keep the zero terminal speed, i.e., $\delta \boldsymbol{v}^q_{k+N|k} = 0$, the perturbation δU^q_k must lie in the null space of $C_{\text{vel}}\check{A}$. By introducing a new vector η, this can be written as

$$\delta U^q_k = E\eta$$

where E denotes an orthonormal basis for the null space of $C_{\text{vel}}\check{A}$. With this δU^q_k, the perturbation to the terminal position is written as

$$\delta \boldsymbol{r}^q_{k+N|k} = C_{\text{pos}}\check{A}E\eta$$

so that η that changes the terminal position $\boldsymbol{r}^q_{k+N|k}$ without changing the terminal velocity $\boldsymbol{v}^q_{k+N|k}$ is parameterized as

$$\eta = (C_{\text{pos}}\check{A}E)^{\mathrm{T}}\big((C_{\text{pos}}\check{A}E)(C_{\text{pos}}\check{A}E)^{\mathrm{T}} \big)^{-1}\xi^q$$

which gives

$$\delta U^q_k = E(C_{\text{pos}}\check{A}E)^{\mathrm{T}}\big((C_{\text{pos}}\check{A}E)(C_{\text{pos}}\check{A}E)^{\mathrm{T}} \big)^{-1}\xi^q \triangleq T^q\xi^q. \tag{7.10}$$

Note that ξ^q in (7.10) has the same dimension as the position, whereas the dimension of ξ^q in (7.9) is n_x.

Then, vehicle p's subproblem P^p_k is written as

$$\min_{\xi^1,\ldots,\xi^{p-1},U^p_k,\xi^{p+1},\ldots,\xi^n} J$$

s.t. (7.4), (7.9) or (7.10),

Algorithm 7.1 Cooperative DRSBK

1: Find a feasible solution of the DRSBK optimization starting from the initial states $\boldsymbol{x}_0^p$, communicate the solutions, form $\bar{U}_0^1, \ldots, \bar{U}_0^n$, and set $k = 1$.
2: **for** $k = 1$ to $k = \infty$ **do**
3: Form the candidate control $\bar{U}_k^1, \ldots, \bar{U}_k^n$ from the previous solution $\bar{U}_{k-1}^1, \ldots, \bar{U}_{k-1}^n$ using (7.12).
4: **for** $p = 1, \ldots, n$ **do**
5: Measure the current states and form $\boldsymbol{x}_k^p$.
6: Solve the subproblem P_k^p.
7: Update the control $\bar{U}_k^p := U_k^{p*}$, $\bar{U}_k^q := \bar{U}_k^q + T^q \xi^{q*}$.
8: Send the solutions to the next vehicle.
9: **end for**
10: Apply the first step of the control inputs $\boldsymbol{u}_k^p = \bar{\boldsymbol{u}}_{k|k}^p$ to the system (7.1), $\forall p$.
11: **end for**

$$F\Big(\mathsf{C}^1(\boldsymbol{x}_k^1, \bar{U}_k^1 + \delta U_k^1), \ldots, \mathsf{C}^{p-1}(\boldsymbol{x}_k^{p-1}, \bar{U}_k^{p-1} + \delta U_k^{p-1}),\ \mathsf{C}^p(\boldsymbol{x}_k^p, U_k^p),$$
$$\tilde{\mathsf{C}}^{p+1}(\boldsymbol{x}_{k|k-1}^{p+1}, \bar{U}_k^{p+1} + \delta U_k^{p+1}), \ldots, \tilde{\mathsf{C}}^n(\boldsymbol{x}_{k|k-1}^n, \bar{U}_k^n + \delta U_k^n)\Big) \leq 0 \qquad (7.11)$$

The full CDRSBK algorithm is shown in Algorithm 7.1. As shown in line 3, when the time step is incremented, the candidate decisions $\bar{U}_k^p$ to be made at the current time are constructed from the decisions $\bar{U}_{k-1}^p$ made at the previous time using

$$\forall p: \qquad \bar{\boldsymbol{u}}_{k+j|k}^p = \bar{\boldsymbol{u}}_{k+j|k-1}^p, \qquad \forall j^- \qquad (7.12a)$$
$$\bar{\boldsymbol{u}}_{k+N-1|k}^p = \kappa^p(\bar{\boldsymbol{x}}_{k+N-1|k-1}^p). \qquad (7.12b)$$

This operation shifts the plan $\bar{U}_{k-1}^p$ by one time step and appends the terminal step taken from the invariance control law $\kappa^p(\boldsymbol{x})$ in (5.26). The states $\bar{\boldsymbol{x}}_{k+N-1|k-1}^p$ are obtained through the state equation

$$\bar{\boldsymbol{x}}_{k+j+1|k-1}^p = A^p \bar{\boldsymbol{x}}_{k+j|k-1}^p + B^p \bar{\boldsymbol{u}}_{k+j|k-1}^p, \qquad \forall j.$$

All vehicles are assumed to have a synchronized discrete time, which is sampled at the time when the control is executed in line 10 of Algorithm 7.1. Non-zero computation and communication times are readily handled by propagating the measured states when forming $\boldsymbol{x}_k^p$ in line 5. Figure 7-1 shows the time lines of the CDRSBK

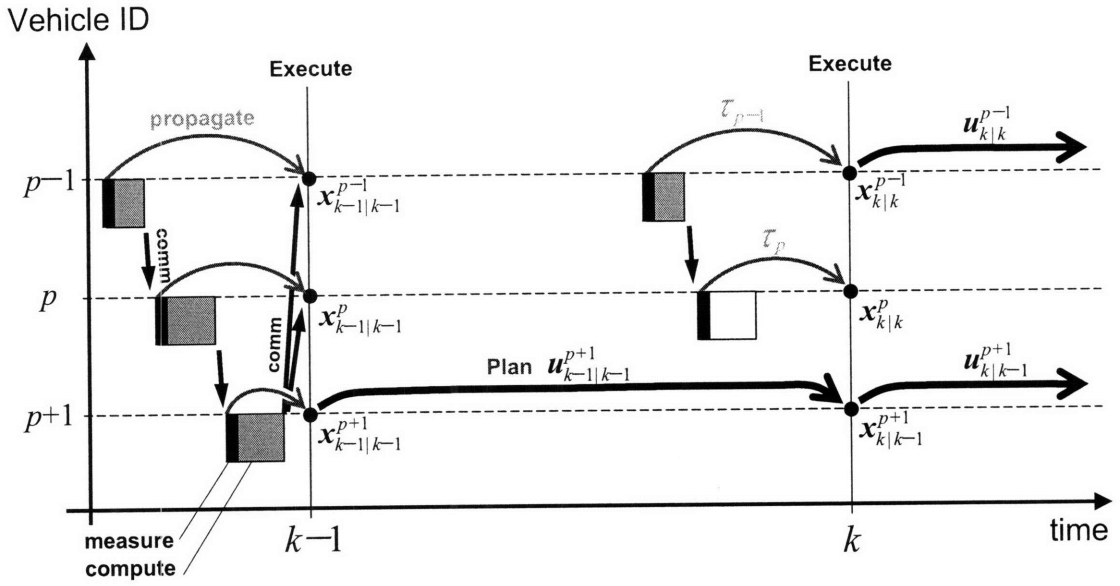


Figure 7-1: Time flow of CDRSBK algorithm with non-zero computation and communication time

algorithm implementation. After receiving the latest plans from the previous vehicle, the vehicle takes a measurement of the latest states immediately before solving its subproblem. The vehicle propagates the measured states up to the discrete execution time, as shown with the gray arrow. When the last vehicle finishes its computation, it broadcasts the final plans to the fleet, and all vehicles implement the control at the same time. Note that when the vehicle p is planning at time k, the prediction for vehicle $p-1$ is based on the latest states $\boldsymbol{x}^{p-1}_{k|k}$ and the inputs $\boldsymbol{u}^{p-1}_{\cdot|k}$, but the prediction for vehicle $p+1$ is based on the states $\boldsymbol{x}^{p+1}_{k|k-1}$ and the inputs $\boldsymbol{u}^{p-1}_{\cdot|k-1}$ that are predicted at the previous time $k-1$.

By aligning the states and the control inputs of all vehicles at the discrete time steps, this framework compiles various sources of uncertainties into one and allows us to treat them as a single prediction error, which can be calculated simply as $\boldsymbol{w}^{p}_{k-1} = \boldsymbol{x}^{p}_{k|k} - \boldsymbol{x}^{p}_{k|k-1}$. Future work will extend this implementation to reduce the delay associated with the propagation, as studied in [52].

7.3.3 MILP Implementation

The following describes the MILP implementation of the CDRSBK algorithm.

Constraints

Output constraints are composed of the following.

- Obstacle Avoidance: (5.51)
- Speed and input constraints: (5.52) and (5.53).
- Invariance Constraints: (5.54), (5.55), and (5.56)
- Interconnected Constraints: (5.57) and (5.58)

For vehicles $q\,(>p)$, the subscript j of the constraint tightening parameters $\alpha_j, \beta_j, \gamma_j$ must be replaced with $j+1$, as shown in (7.5)–(7.6).

Objective Function

The individual objective function is the same as (5.59)–(5.60) except that the cost-to-go is evaluated from a point $\boldsymbol{O}^p$ in the invariant set $\mathcal{S}_k^p$

$$J^p = \|\boldsymbol{O}^p - \boldsymbol{r}^p_{\text{vis}}\|_2 + \tilde{f}^p(\boldsymbol{r}^p_{\text{vis}}).$$

The MILP implementation of the fleet cost (7.4) is

$$\min J_{\text{worst}} + \epsilon \sum_{p=1}^{n} J^p \tag{7.13a}$$

$$J_{\text{worst}} \geq J^p, \qquad \forall p \tag{7.13b}$$

$$J^p \geq [\cos\theta_m, \sin\theta_m](\boldsymbol{O}^p - \boldsymbol{r}^p_{\text{vis}}) + \tilde{f}^p(\boldsymbol{r}^p_{\text{vis}}), \qquad \forall m \tag{7.13c}$$

(5.59).

To ensure visibility of the cost point $\boldsymbol{r}^p_{\text{vis}}$ from $\boldsymbol{O}^p$, obstacle avoidance constraints are enforced on n_{int} interpolation points placed on the line connecting $\boldsymbol{r}^p_{\text{vis}}$ and $\boldsymbol{O}^p =$

$[x^p_{\text{center}},\ y^p_{\text{center}}]^{\mathrm{T}}$

$$\mu_l x^p_{\text{center}} + (1-\mu_l) x^p_{\text{vis}} \le x_{\text{low},o} - \alpha_{N-1} + M\, b^p_{\text{int},lo1} \tag{7.14a}$$

$$\mu_l y^p_{\text{center}} + (1-\mu_l) y^p_{\text{vis}} \le y_{\text{low},o} - \alpha_{N-1} + M\, b^p_{\text{int},lo2} \tag{7.14b}$$

$$\mu_l x^p_{\text{center}} + (1-\mu_l) x^p_{\text{vis}} \ge x_{\text{high},o} + \alpha_{N-1} - M\, b^p_{\text{int},lo3} \tag{7.14c}$$

$$\mu_l y^p_{\text{center}} + (1-\mu_l) y^p_{\text{vis}} \ge y_{\text{high},o} + \alpha_{N-1} - M\, b^p_{\text{int},lo4} \tag{7.14d}$$

$$\sum_{i=1}^{4} b^p_{\text{int},loi} \le 3 \tag{7.14e}$$

$$\mu_l = \frac{l}{n_{\text{int}}}, \qquad l = 1, \ldots, n_{\text{int}}.$$

Decision Variables

The inputs of other vehicles are the sum of the published inputs $\bar{U}^q_k$ and the parameterized perturbation

$$\forall q: \qquad \begin{bmatrix} \boldsymbol{u}^q_{k|k} \\ \vdots \\ \boldsymbol{u}^q_{k+N-1|k} \end{bmatrix} = \bar{U}^q_k + T^q \xi^q. \tag{7.15}$$

The binary variable b^q_{left} selects the left/right safety circle, and the location of the safety circle has a large effect on the objective value. Therefore it is beneficial to keep the binary b^q_{left} as a free decision variable in p's optimization, as mentioned in Section 7.3.2. Other binary variables are fixed if the superscript does not include p.

$$\forall q: \qquad b^q_{\text{obst}} = \bar{b}^q_{\text{obst}}, \tag{7.16a}$$

$$b^q_{\text{circ-obst}} = \bar{b}^q_{\text{circ-obst}}, \tag{7.16b}$$

$$b^q_{\text{vis}} = \bar{b}^q_{\text{vis}}, \tag{7.16c}$$

$$b^q_{\text{int}} = \bar{b}^q_{\text{int}}, \tag{7.16d}$$

$$b^{qr}_{\text{veh}} = \bar{b}^{qr}_{\text{veh}}, \quad \forall r \ne p,\ r \ne q \tag{7.16e}$$

$$b^{qr}_{\text{circ-circ}} = \bar{b}^{qr}_{\text{circ-circ}}, \quad \forall r \ne p,\ r \ne q \tag{7.16f}$$

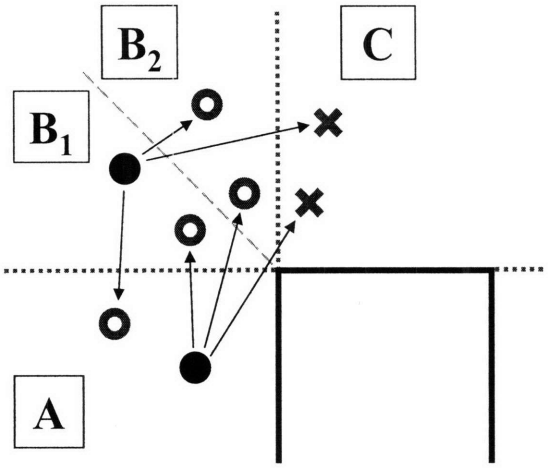


Figure 7-2: Feasible perturbation of the position, with fixed obstacle avoidance binaries

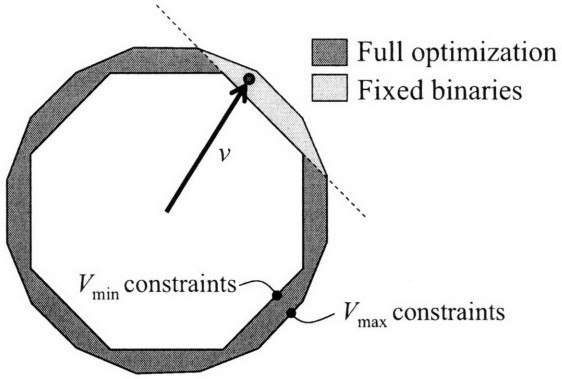


Figure 7-3: Feasible perturbation of the velocity, with fixed $v_{\min}$ binaries

In summary, the MILP implementation of subproblem P_k^p is to minimize (7.13a) subject to the initial condition (5.9), state prediction (5.10), constraints (5.51)–(5.59), and (7.13b)–(7.16).

7.3.4 Effect of the Fixed Binary Variables

The CDRSBK algorithm fixes binaries of other vehicles, while solving for perturbations of their continuous variables. Figure 7-2 shows the effect of fixed binaries on the obstacle avoidance (5.51). In region **A**, the admissible binaries are [0, 1, 1, 1] where 1 means the avoidance constraint is relaxed. In region **C**, the binaries are [1, 1, 1, 0]. The regions $\mathbf{B}_1$, $\mathbf{B}_2$ have three possible binary settings [0, 1, 1, 1], [1, 1, 1, 0], and [0, 1, 1, 0], and the output of the MILP solver could be any of them. If the binaries are fixed [0, 1, 1, 0] for a point in region **B**, which is a union of $\mathbf{B}_1$ and $\mathbf{B}_2$, then the perturbed point must also stay inside the region **B** only. To enable larger perturbations, CDRSBK algorithm performs a MILP pre-processing and sets [0, 1, 1, 1] as the binaries of the point in $\mathbf{B}_1$, which allows the point to move within **A**, $\mathbf{B}_1$, and $\mathbf{B}_2$. Similarly, the point in $\mathbf{B}_2$ uses the binary setting [1, 1, 1, 0]. As an illustration of this binary fix, Figure 7-2 shows feasible and infeasible perturbations from a point • with ∘ and × respectively. The binaries of the non-convex minimum speed constraints are

fixed in the same way to allow for maximum perturbation, as shown in Figure 7-3.

Note that the problem statement includes many binary variables that are effectively fixed by (7.16) or constraints that are always satisfied (e.g. the lower left side of constraints in Figure 7-3). The MILP solver CPLEX eliminates these redundant variables and constraints in the pre-solve step, and the size of the CDRSBK subproblem increases only slightly from the DRSBK subproblem. In the example shown later in Section 7.5, the number of variables after the CPLEX pre-solve increased by 15% for continuous variables and 1% for binary variables.

7.4 Algorithm Properties

This section discusses the two important properties of CDRSBK algorithm in terms of constraint satisfaction and the performance.

7.4.1 Robust Feasibility

The next theorem guarantees the robust feasibility of the entire fleet after solving each subproblem.

Theorem 7.1. *If a feasible solution $\bar{U}_0^1, \ldots, \bar{U}_0^n$ to the following constraint ((7.7) with $k = 0$, $p = n$) is found*

$$F\Big(\mathsf{C}^1(\boldsymbol{x}_0^1, \bar{U}_0^1), \ldots, \mathsf{C}^n(\boldsymbol{x}_0^n, \bar{U}_0^n)\Big) \leq 0 \tag{7.17}$$

then, the system (7.1) *controlled by Algorithm 7.1 satisfies the constraints* (7.2)–(7.3) *under the action of disturbance $\boldsymbol{w}_k^p \in \mathcal{W}^p$ for all vehicles p and all future times k (> 0).*

Proof. The proof is based on a recursion. The argument is very similar to the proofs in Chapter 5.

First, show that vehicle 1's optimization at time k (> 0) is feasible, assuming the feasibility of vehicle n's optimization at time $k-1$. After vehicle n updates the control

input in line 7 of Algorithm 7.1, we have

$$F\big(\mathsf{C}^1(\boldsymbol{x}^1_{k-1}, \bar{U}^1_{k-1}), \ldots, \mathsf{C}^n(\boldsymbol{x}^n_{k-1}, \bar{U}^n_{k-1})\big) \leq 0 \tag{7.18}$$

From this solution $\bar{U}^1_{k-1}, \ldots, \bar{U}^n_{k-1}$ to (7.18), one can construct a candidate control $\bar{U}^1_k, \ldots, \bar{U}^n_k$ by shifting the solution by one time step and appending a terminal control at the end, as shown in (7.12). Because the constraints $\tilde{\mathsf{C}}^p(\boldsymbol{x}^p_{k|k-1}, \bar{U}^p_k)$ are constructed also by shifting $\mathsf{C}^p(\boldsymbol{x}^p_{k-1}, \bar{U}^p_{k-1})$ by one time step and the starting states $\boldsymbol{x}^p_{k|k-1}$ of $\tilde{\mathsf{C}}^p(\boldsymbol{x}^p_{k|k-1}, \bar{U}^p_k)$ is the second step of $\mathsf{C}^p(\boldsymbol{x}^p_{k-1}, \bar{U}^p_{k-1})$, the candidate control $\bar{U}^1_k, \ldots, \bar{U}^n_k$ satisfies

$$F\big(\tilde{\mathsf{C}}^1(\boldsymbol{x}^1_{k|k-1}, \bar{U}^1_k), \ldots, \tilde{\mathsf{C}}^n(\boldsymbol{x}^n_{k|k-1}, \bar{U}^n_k)\big) \leq 0 \tag{7.19}$$

which is a shifted version of (7.18). Using the following

$$\hat{\boldsymbol{u}}^1_{k+j|k} = \bar{\boldsymbol{u}}^1_{k+j|k-1} + P^1_{j+1}\boldsymbol{w}^1_{k-1}, \quad \forall j^- \tag{7.20a}$$

$$\hat{\boldsymbol{x}}^1_{k+j|k} = \bar{\boldsymbol{x}}^1_{k+j|k-1} + L^1_j\boldsymbol{w}^1_{k-1}, \quad \forall j \tag{7.20b}$$

$$\hat{\boldsymbol{u}}^1_{k+N-1|k} = \kappa^1(\hat{\boldsymbol{x}}^1_{k+N-1|k}), \tag{7.20c}$$

$$\hat{\boldsymbol{x}}^1_{k+N|k} = A^1\hat{\boldsymbol{x}}^1_{k+N-1|k} + B^1\hat{\boldsymbol{u}}^1_{k+N-1|k} \tag{7.20d}$$

for vehicle 1, which is denoted by $\hat{U}^1_k$, and letting

$$\delta U^q_k = 0, \qquad \forall q$$

for other vehicles, it can be shown that (7.19) implies

$$F\big(\mathsf{C}^1(\boldsymbol{x}^1_k, \hat{U}^1_k), \tilde{\mathsf{C}}^2(\boldsymbol{x}^2_{k|k-1}, \bar{U}^2_k), \ldots, \tilde{\mathsf{C}}^n(\boldsymbol{x}^n_{k|k-1}, \bar{U}^n_k)\big) \leq 0 \tag{7.21}$$

for any $\boldsymbol{w}^1_{k-1}$, by comparing $\tilde{\mathsf{C}}^1(\boldsymbol{x}^1_{k|k-1}, \bar{U}^1_k)$ and $\mathsf{C}^1(\boldsymbol{x}^1_k, \hat{U}^1_k)$. This shows that the feasibility of the last vehicle's ($p = n$) optimization at time $k-1$ guarantees the feasibility of the first vehicle's optimization at the next time k.

Second, show that vehicle p's optimization at time k is feasible, assuming the feasibility of $(p-1)$'s optimization at time k. After the vehicle $p-1$ updates the control input in line 7 of Algorithm 7.1,

$$F\big(\mathsf{C}^1(\boldsymbol{x}_k^1, \bar{U}_k^1), \ldots, \mathsf{C}^{p-1}(\boldsymbol{x}_k^{p-1}, \bar{U}_k^{p-1}), \tilde{\mathsf{C}}^p(\boldsymbol{x}_{k|k-1}^p, \bar{U}_k^p), \ldots, \tilde{\mathsf{C}}^n(\boldsymbol{x}_{k|k-1}^n, \bar{U}_k^n)\big) \leq 0.$$

The set of constraints for p's optimization is

$$F\big(\mathsf{C}^1(\boldsymbol{x}_k^1, \bar{U}_k^1), \ldots, \mathsf{C}^p(\boldsymbol{x}_k^p, \bar{U}_k^p), \tilde{\mathsf{C}}^{p+1}(\boldsymbol{x}_{k|k-1}^{p+1}, \bar{U}_k^{p+1}), \ldots, \tilde{\mathsf{C}}^n(\boldsymbol{x}_{k|k-1}^n, \bar{U}_k^n)\big) \leq 0 \quad (7.22)$$

and is the same set of constraints as $(p-1)$'s, except for the change from $\tilde{\mathsf{C}}^p(\boldsymbol{x}_{k|k-1}^p, \bar{U}_k^p)$ to $\mathsf{C}^p(\boldsymbol{x}_k^p, U_k^p)$. It can be shown that p's optimization has the following candidate solution

$$\hat{\boldsymbol{u}}_{k+j|k}^p = \bar{\boldsymbol{u}}_{k+j|k-1}^p + P_{j+1}^p \boldsymbol{w}_{k-1}^p, \quad \forall j^- \quad (7.23a)$$

$$\hat{\boldsymbol{x}}_{k+j|k}^p = \bar{\boldsymbol{x}}_{k+j|k-1}^p + L_j^p \boldsymbol{w}_{k-1}^p, \quad \forall j \quad (7.23b)$$

$$\hat{\boldsymbol{u}}_{k+N-1|k}^p = \kappa^p(\hat{\boldsymbol{x}}_{k+N-1|k}^p), \quad (7.23c)$$

$$\hat{\boldsymbol{x}}_{k+N|k}^p = A^p \hat{\boldsymbol{x}}_{k+N-1|k}^p + B^p \hat{\boldsymbol{u}}_{k+N-1|k}^p \quad (7.23d)$$

for its own decision and

$$\delta U_k^q = 0, \quad \forall q \quad (7.24)$$

for other vehicles' decisions. This solution is feasible to (7.22) for any disturbance realization $\boldsymbol{w}_{k-1}^p$ that acted on vehicle p at time $k-1$, showing that the feasibility of $(p-1)$'s optimization guarantees the feasibility of p's optimization.

Therefore, with the initial feasibility (7.17), the subproblem remains always feasible. The initial step of the control $\boldsymbol{u}_{k|k}^p$ and the states $\boldsymbol{x}_{k|k}^p$ satisfy the original constraints (7.2)–(7.3), and the robust feasibility of the fleet is proven. □

7.4.2 Monotonic Decrease of the Fleet Cost

Another important property of CDRSBK algorithm is that the fleet objective is monotonically decreasing by solving each subproblem.

Theorem 7.2. *The fleet objective value* (7.4) *monotonically decreases by solving each subproblem* P_k^p *over the fleet (line 4 in Algorithm 7.1) and over the time (line 2).*

Proof. The proof is based on showing that the candidate solution to p's optimization yields the objective value that is no worse than the objective value found in the optimization of the previous vehicle $p-1$ (or n if $p=1$).

In p's subproblem, the candidate solution (7.24) does not change variables for vehicles $\forall q \neq p$, and hence the local cost J^q. For vehicle p, with the assumption of nilpotency $L_{N-1}^p = 0$, the terminal states can be written as

$$\hat{\boldsymbol{x}}_{k+N|k}^p = A^p \bar{\boldsymbol{x}}_{k+N-1|k-1}^p + B^p \kappa^p(\bar{\boldsymbol{x}}_{k+N-1|k-1}^p)$$

using (7.23b)–(7.23d). If the terminal controller $\kappa^p(\cdot)$ makes the vehicle loiter in a circle with a constant turning radius, the center of the invariant set $\boldsymbol{O}^p$ does not move. This means the individual objective value J^p of the candidate solution does not change. Because p's optimization can use the candidate solution to achieve the same fleet cost updated by the previous vehicle $(p-1)$, the fleet cost can only improve by optimization. □

Note that this does not mean the monotonic decrease of the individual cost. The simulation results in Section 7.5 demonstrate a temporary increase of the individual cost that leads to a greater reduction of the fleet cost.

7.5 Simulation Results

This section presents the performance comparison of DRSBK and CDRSBK through simulation.

7.5.1 Setup

The simulation uses homogeneous fixed-wing UAVs, whose dynamics are described in Section 5.5.1. The disturbance magnitude $w_{\max}$ is 5% of the control authority $a_{\max}$. The planning horizon length N is 4. The parameters for dynamic constraints are: $v_{\min} = 18$, $v_{\max} = 24$, $a_{\max} = 3.84$. A two-step nilpotent controller is used for this system, and the parameters for constraint tightening are obtained through (5.50)

$$\begin{aligned} &\alpha_0 = 0, && \beta_0 = 0, && \gamma_0 = 0, \\ &\alpha_1 = 2.4, && \beta_1 = 1.4, && \gamma_1 = 0.54, \\ &\alpha_j = 4.8, && \beta_j = 2.7, && \gamma_j = 0.81, \quad j \geq 2. \end{aligned}$$

The parameterization matrix T^q is calculated from (7.9)

$$T^q = \begin{bmatrix} 0.012 & 0 & -0.07 & 0 \\ 0 & 0.012 & 0 & -0.07 \\ 0.004 & 0 & 0.01 & 0 \\ 0 & 0.004 & 0 & 0.01 \\ -0.004 & 0 & 0.09 & 0 \\ 0 & -0.004 & 0 & 0.09 \\ -0.012 & 0 & 0.17 & 0 \\ 0 & -0.012 & 0 & 0.17 \end{bmatrix}.$$

7.5.2 Results

The scenario considers two vehicles trying to reach their own targets (marked with $\square$) while avoiding obstacles and the other vehicle. The goal is to minimize the mission completion time with a small penalty $\epsilon = 0.05$ on the individual cost in (7.4). Figures 7-4 and 7-6 show the trajectories generated by DRSBK and CDRSBK algorithm respectively. The trajectory of vehicle 1 is marked with $\times$, and that of vehicle 2 is marked with $\triangle$. Because vehicle 1 has to traverse a longer route, the optimal solution is for vehicle 2 to back off. Figures 7-5 and 7-7 show the trajectories optimized in

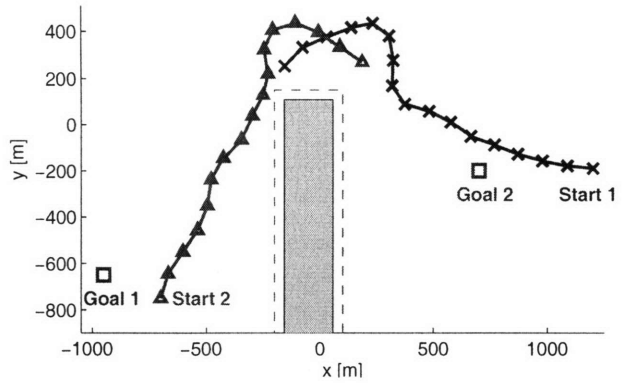


Figure 7-4: Trajectories executed using DRSBK algorithm

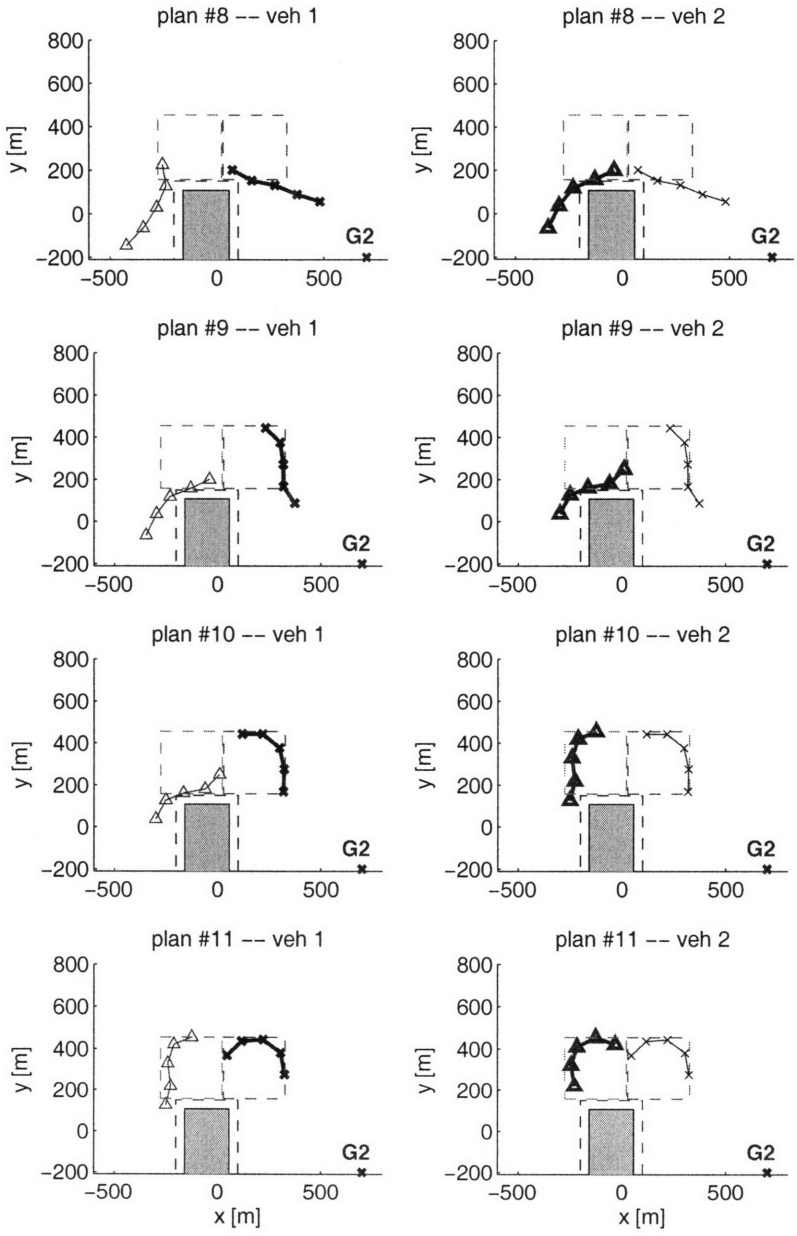


Figure 7-5: Trajectories generated by DRSBK algorithm at each time step

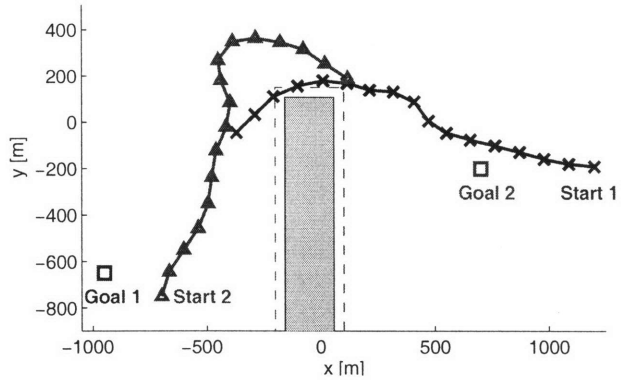


Figure 7-6: Trajectories executed using CDRSBK algorithm

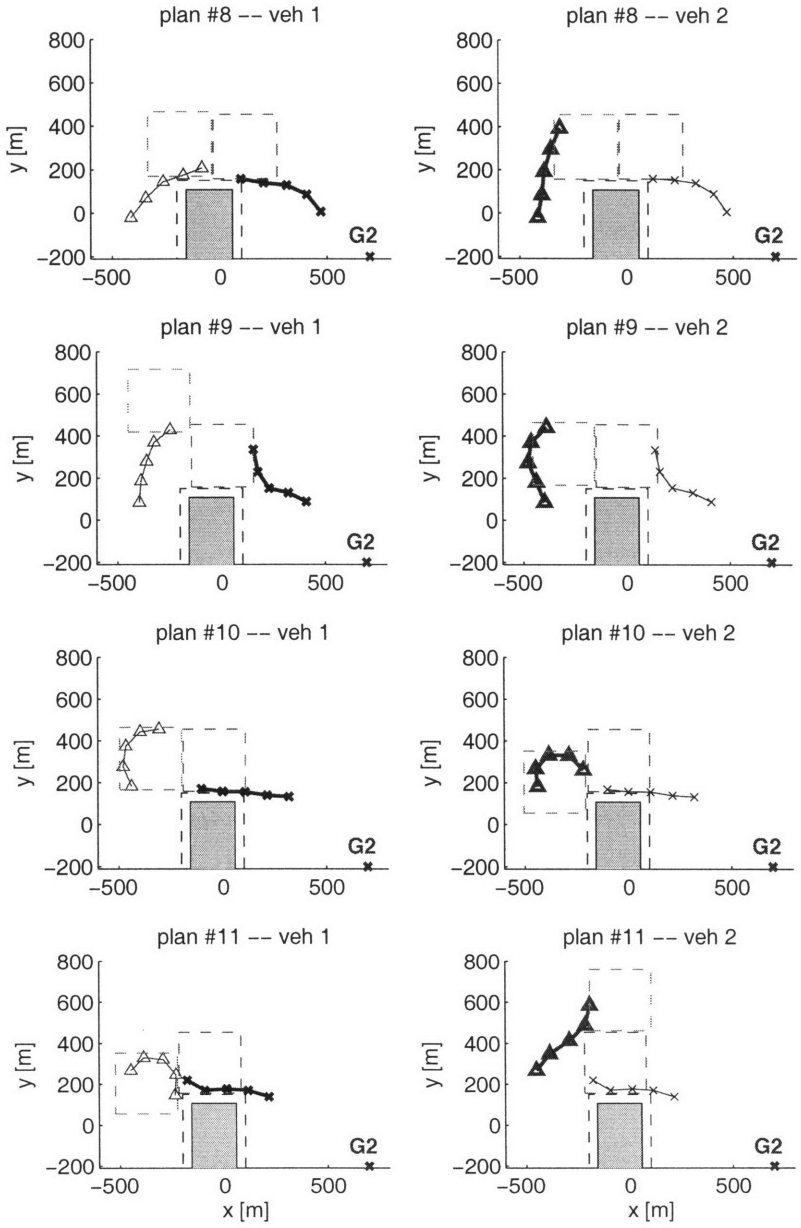


Figure 7-7: Trajectories generated by CDRSBK algorithm at each time step

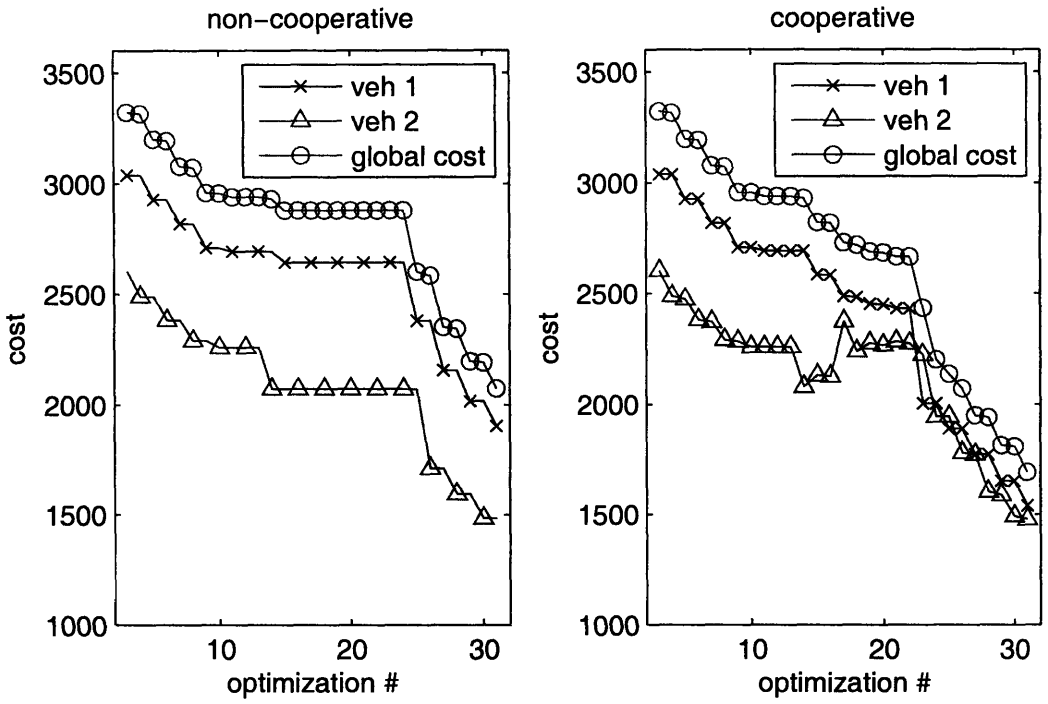


Figure 7-8: Time history of the objective function

each subproblem from time $k = 8$ to 11. The trajectory of the vehicle solving the subproblem is shown in thick lines. The dashed boxes show the region where the safety circle lie in.

Both distributed algorithms maintained feasibility under the action of disturbances. As shown in Figure 7-5, however, DRSBK subproblem solely minimizes the individual cost without accounting for the performance of the other vehicle, making no improvements on the cost for some time. In CDRSBK, plan #9 of vehicle 1 flips the side of loiter circle of vehicle 2. Furthermore, in plan #9, vehicle 2 sacrifices its own objective for the improvement of vehicle 1's cost, by moving the trajectory towards the left of the figure.

This cooperative behavior is seen also in the objective values. Figure 7-8 shows the time history of the individual cost J^p and the fleet cost J. Both algorithms monotonically decrease the fleet objective. As shown in the right figure, the cooperative formulation allows the individual cost to increase if it leads to a larger improvement of the fleet cost. Between optimization #14–17 (which correspond to time 7–9), the

vehicle with a better cost (vehicle 2) yields to the vehicle with a worse cost (vehicle 1), enabling a large reduction in the fleet cost J. The average computation time for solving MILP in this scenario was 0.050 second for DRSBK and 0.064 second for CDRSBK.

7.6 Hardware Results

A similar scenario is tested with two quadrotors introduce in Section 4.4.1. A planning laptop is assigned to each vehicle, and the inter-vehicle communication is implemented as a communication over the TCP/IP network. The following parameters were used in the algorithm.

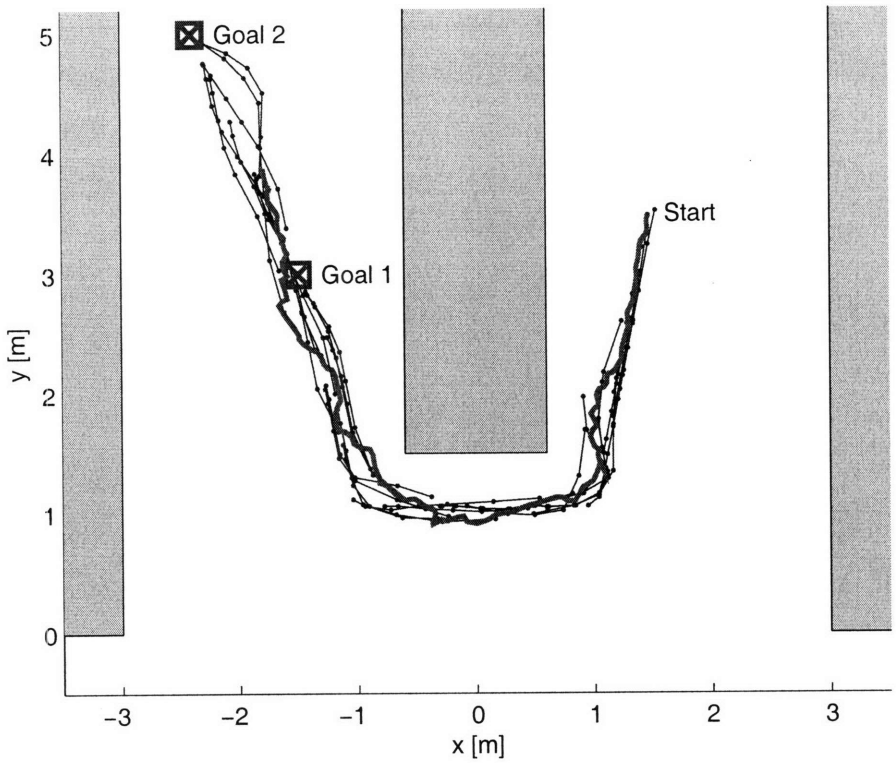
$$\begin{aligned} \Delta t &= 2.5\,\text{sec}, & N &= 6 \\ \tau_1 &= 1.3\,\text{sec}, & \tau_2 &= 0.7\,\text{sec} \\ v_{\max} &= 0.30\,\text{m/s}, & a_{\max} &= 0.45\,\text{m/s}^2 \\ w_r &= 0.27\,\text{m}, & w_v &= 0.09\,\text{m/s} \end{aligned}$$

where τ_1 and τ_2 are the propagation time for vehicle 1 and 2, as shown in Figure 7-1. The offline procedure in Chapter 3 produced the following constraint contraction.

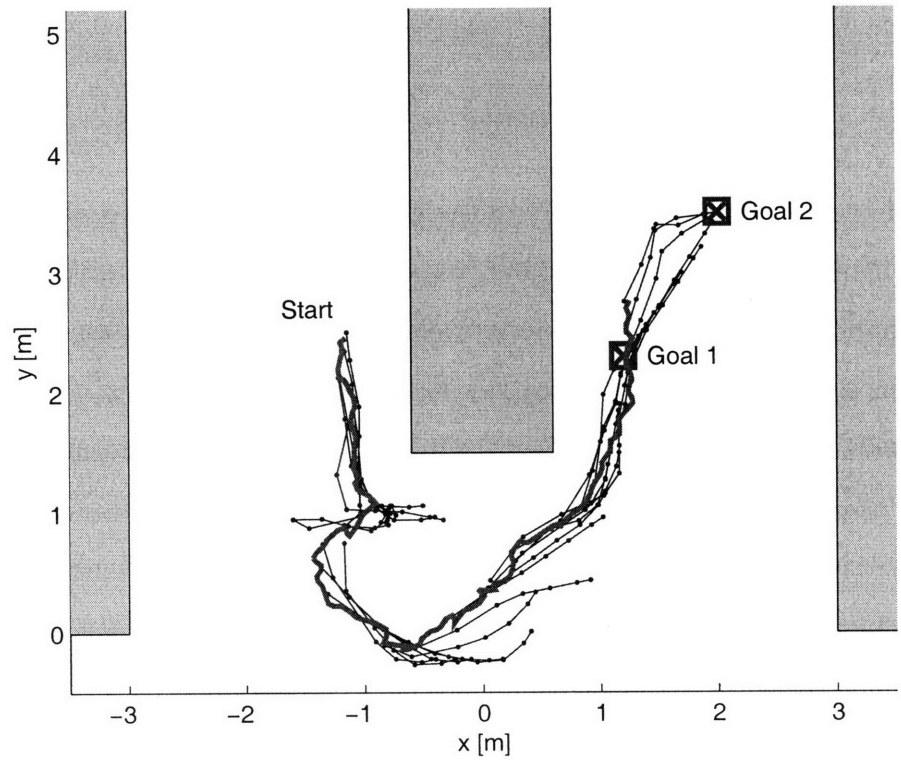
$$\begin{aligned} \alpha_0 &= 0, & \beta_0 &= 0, & \gamma_0 &= 0, & \\ \alpha_1 &= 0.27, & \beta_1 &= 0.09, & \gamma_1 &= 0.067, & \\ \alpha_2 &= 0.428, & \beta_2 &= 0.198, & \gamma_2 &= 0.106, & \\ \alpha_j &= 0.440, & \beta_j &= 0.208, & \gamma_j &= 0.110, & j \geq 3 \end{aligned}$$

The vehicles 1 and 2 started around (1.5, 3.5) and (−1.2, 2.5) respectively. The targets for vehicle 1 are (−1.5, 3.0) are (−2.4, 5.0), and the targets for vehicle 2 are $(1.2, 2.3)$ and $(2.0, 3.5)$, which are all marked with ⊠. The vehicles must switch the position while avoiding the other vehicle and the obstacle in the middle.

Figure 7-9 shows the scenario and the plot of the trajectories of each vehicle. The



(a) Vehicle 1



(b) Vehicle 2

Figure 7-9: The plans and the trajectories of two quadrotors from the CDRSBK algorithm experiment

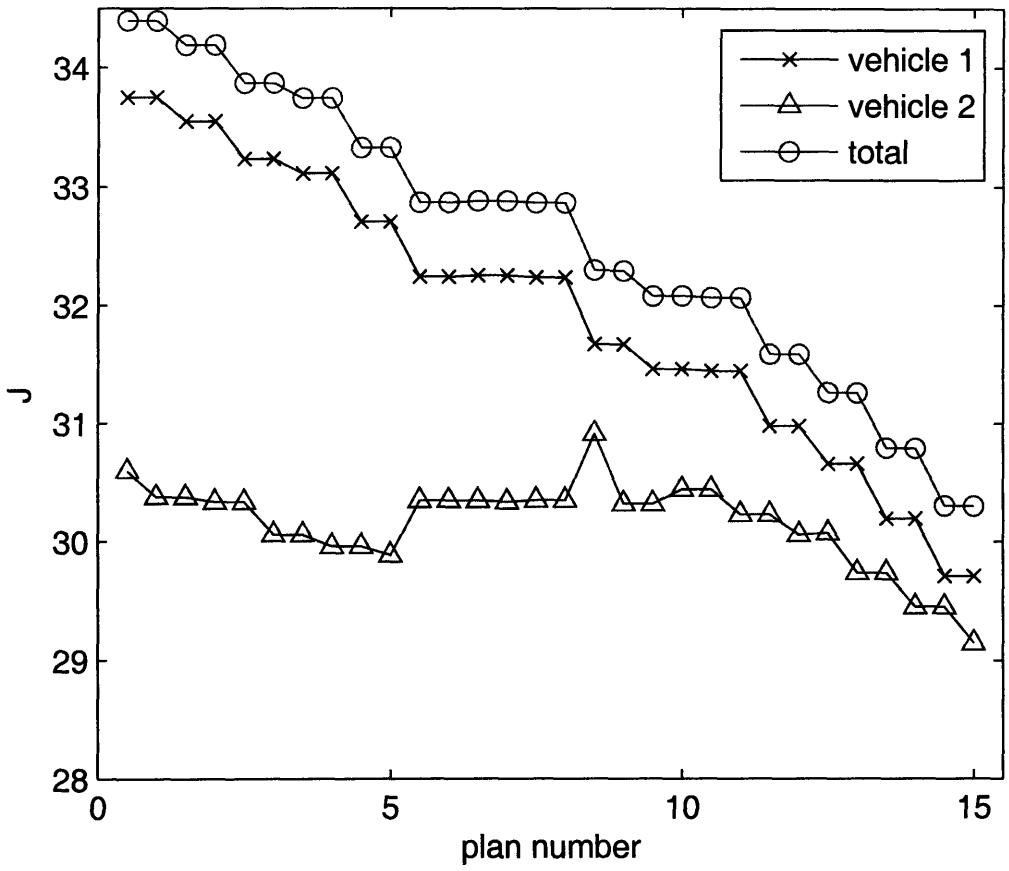


Figure 7-10: Individual costs and the global cost

red thick line is the actual trajectory of the vehicle, which were recorded at 2 Hz. All the plans generated in the receding horizon framework are shown with blue lines.

As a result of their initial locations, vehicle 2 approaches the bottom of the obstacle before vehicle 1, and vehicle 2 then tries to go along the bottom of the obstacle, as the planned trajectories in Figure 7-9(b) show. This would have the effect of delaying vehicle 1, which has targets that are further away, and already has a longer mission to execute than vehicle 2. Thus, vehicle 2 yields way to vehicle 1 to minimize the fleet mission completion time in (7.4). This cooperative effect is also shown in Figure 7-10, which plots the objective values of the first 15 plans. Note that between plans 5 and 9, there is a temporary increase of the cost for vehicle 2, but, even under the action of disturbances, the total objective value is monotonically decreasing. The average computation time for each local optimization on a 2.4GHz laptop was 0.31 second.

This flight test successfully demonstrated the cooperative behavior by the distributed online planning algorithm.

7.7 Summary

This chapter extended a robust decentralized trajectory optimization algorithm to include explicit cooperation. Each vehicle sequentially solves the subproblem, but the subproblem also includes the global objective and feasible modifications to other vehicles' plans. In order to maintain the scalability of the algorithm, continuous variables of other vehicles are parameterized using a variable of smaller dimension, while most binary variables are fixed. The result shows the robust feasibility and the monotonic decrease of the global cost with a marginal increase in the computation.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

Receding Horizon MILP (Chapter 2)

Chapter 2 has extended the previous RH-MILP formulation to three dimensions. In the cost map construction phase, an LP formulation is presented that can be applied to the visibility check of the general n-dimension case. In the MILP optimization phase, a new objective function is developed to enable the vehicle to stay close to the ground but fly over obstacles if necessary. Depending on the penalty of the altitude, the approach is shown to generate various different trajectories in complex scenarios. Numerical simulation also demonstrated the reduction of the computation time by using an initial guess for MILP within CPLEX.

Constraint Tightening Robust MPC (Chapter 3)

Chapter 3 reparameterized the feedback correction used in the constraint tightening robust MPC, and showed that the new approach can represent a larger class of disturbance rejection policies. In order to address the question of how to design a good feedback correction off-line, the chapter presented the necessary and sufficient conditions for the existence of a feasible set. Using these conditions, it then presented an off-line linear optimization to design the disturbance rejection controller that max-

imizes the disturbance level that the controller can handle. The simulation results demonstrated significant performance improvements over the previous approaches at high disturbance levels.

Robust Receding Horizon Trajectory Optimization (Chapter 4)

Chapter 4 combined the RH-MILP and the robust MPC algorithm while retaining the advantages of both algorithms, i.e., the algorithm is robust to disturbances, safe against environmental changes, and able to generate a knowledgeable trajectory using a short planning horizon. The robust stability of the planner was proven by showing that the candidate solution gives a cost that is no worse than the current cost and hence the cost is monotonically decreasing. The hardware experiments using the quadrotor testbed demonstrated that the algorithm can fly the vehicle in a three dimensional environment close to the constraint boundary, while robustly accounting for the prediction errors arising from various sources including wind disturbances, the modeling error of the vehicle, and a tracking error of the low-level waypoint follower.

Distributed RSBK Algorithm (Chapter 5)

Chapter 5 first extended the RSBK algorithm in Chapter 4 to the multi-vehicle case. Then, a decentralized form is developed by having each vehicle solve a local subproblem, whose problem complexity is much smaller than the centralized problem. This reduces the computation of the vehicle team without a significant increase in the communication. Because the algorithm uses a short planning horizon, the initialization of the distributed algorithm is much simpler than the previous work, so that it is better suited for online planning. The robust feasibility of the entire fleet with local planning and local communication is proven. A grouping algorithm is also integrated to enable simultaneous computation among the vehicle team. The overall distributed planning architecture was implemented on the multi-rover testbed and the results of the hardware experiments demonstrated the onboard and online computation capabilities of the algorithm as well as the robust constraint satisfaction in the real world.

Cooperative Decentralized Trajectory Optimization (Chapter 6 and 7)

The last part of the thesis focused on how to improve the team performance and achieve a cooperative behavior by solving local optimizations. Chapter 6 developed a decentralized cooperation algorithm that enables vehicles to sacrifice the local cost if it leads to a larger improvement of the global cost. The key was to parameterize the decision of other vehicles using the active coupling constraints. The approach is shown to be more scalable than the centralized algorithm and gives better performance than the non-cooperative distributed algorithm. Using this technique, Chapter 7 extended the DRSBK algorithm, where each vehicle optimizes its local decision as well as a feasible modification to the prediction of other vehicles' plans. The overall robust cooperative distributed algorithm was demonstrated on the multi-quadrotor testbed, showing the validity of the computation and communication requirements of this algorithm.

8.2 Future Research Directions

Building on the contributions listed above, suggestions for the future research directions are outlined below.

Unification/comparison of Robust MPC

Much of the robust MPC work [41, 90, 100, 121] rely on similar underlying concepts in order to ensure the robustness and the stability of the controller. It would be valuable to put together different formulations or different notations into one framework and compare the approaches in terms of performance, computation complexity, region of feasibility, and the degradation of these with respect to the disturbance level.

Bounded Disturbance with Time Correlation

The robust MPC algorithms presented in this thesis ensured that the controller is robust to any disturbance realization from a bounded set. In reality, it is unlikely that

the worst case disturbance keeps acting on the system over a long time. An alternative formulation could be developed that captures the time correlation of the disturbance sequence. This will exploit the previous history of disturbances in predicting the future disturbance realization, and could reduce the conservatism of the controller.

Analysis of Performance Robustness

Multi-parametric programming [99] is a tool that can calculate off-line an explicit solution to LP, QP, and MILP. The explicit solution of the optimal controller is expressed in the form of a set of partitioned state space, where each partition has a constant control gain. To obtain the control input, the original online optimization becomes an online evaluation of which partition the current states are in. Further analysis will enable us to calculate the expected and/or worst-case performance of the complicated optimization-based controllers under the presence of disturbances. This could be extended to identify other key issues such as noise sensitivity, controller bandwidth, robustness margin, and the effect of filtering on the overall performance.

Control of the Closed-loop System

With the increase of the autonomy of the unmanned vehicle systems, the system typically has multiple control loop closures, such as low-level control, guidance, and task allocation. Since each control loop focuses on a different aspect of the problem, they can be designed separately in the ideal situation. For example, when designing the outer-loop MPC, the inner-loop controller is assumed to be either given or chosen by trial and error [122]. However, in order to enhance the capability of the overall system, tighter integration of the multiple loops is desired. With the performance analysis tool of the MPC controller, one can investigate the effect of the low-level controller design on the higher-level controllers and can ensure the intention of the inner loop controller matches that of the high-level controller. This could enable a new simultaneous design procedure of inner and outer control loops.

Communication Modeling for Distributed Planning

The communication in the distributed planning system introduces continuous delays and discrete packet loss. More detailed modeling of the communication network will allow the planner to explicitly account for the stochastic nature of the inter-vehicle communication. Further work is required to extend the existing robust planner to be robust to these different types of uncertainties.

Asynchronous Control of Multiple Vehicles

In this thesis, the vehicles in the fleet are assumed to have the synchronized discrete time step. As the number of vehicles involved in the planning process increases, ensuring the perfect synchronization among the large vehicle fleet becomes difficult. Asynchronous algorithms in the distributed setting have been studied in the field of Computer Science [123], and it would be interesting to expand the UAV research into such a field.

Bibliography

[1] Personal communication with Jim McGrew, a former USAF Captain and Predator Pilot, Dec 2006.

[2] L. R. Newcome. *Unmanned Aviation: A Brief History of Unmanned Aerial Vehcles.* AIAA, 2004.

[3] J. Bay. Heterogeneous Urban RSTA Team (HURT). Technical report, DARPA/IXO, December 2003.

[4] Office of the Secretary of Defense. Unmanned aerial systems roadmap. Technical report, December 2005. URL `http://www.acq.osd.mil/usd/RoadmapFinal2.pdf`.

[5] T. Samad and G. Balas (Eds.). *Software-Enabled Control: Information Technology for Dynamical Systems.* Wiley-IEEE Press, 2003.

[6] P. R. Chanler and M. Pachter. Hierarchical Control for Autonomous Teams. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Montreal, August 2001.

[7] W. Findeisen, F. N. Bailey, M. Brdys, K. Malinowski, and A. Wozniak. *Control and coordination in hierarchical systems*, volume 9. A Wiley – Interscience Publication, London, 1980.

[8] Y. Kuwata. Real-time Trajectory Design for Unmanned Aerial Vehicles using Receding Horizon Control. Master's thesis, Massachusetts Institute of Technology, June 2003.

[9] A. Richards, J. Bellingham, M. Tillerson, and J. How. Coordination and Control of Multiple UAVs. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Monterey, CA, Aug 2002.

[10] M. Alighanbari. Task Assignment Algorithms for Teams of UAVs in Dynamic Environments. Master's thesis, MIT, June 2004.

[11] P. R. Chandler, M. Pachter, D. Swaroop, J. M. Fowler, J. K. Howlett, S. Rasmussen, C. Schumacher, and K. Nygard. Complexity in UAV Cooperative Control. In *Proceedings of the IEEE American Control Conference*, pages 1831–1836, Anchorage AK, May 2002.

[12] M. Moser, D. Jokanovic, and N. Shiratori. An algorithm for the multidimensional multiple-choice knapsack roblem. In *IEICE Trans. Fundamentals E80-A(3)*, page 582589, March 1997.

[13] C. Schumacher, P. R. Chandler, and S. Rasmussen. Task Allocation for Wide Area Search Munitions via Network Flow Optimization. In *Proceedings of the IEEE American Control Conference*, pages 1917–1922, Anchorage AK, May 2002.

[14] E. Frazzoli, M. Dahleh, and E. Feron. Real-Time Motion Planning for Agile Autonomous Vehicles. *AIAA Journal of Guidance, Control, and Dynamics*, 25 (1):116–129, 2002.

[15] T. Schouwenaars, M. Valenti, E. Feron, and J. P. How. Linear programming and language processing for human-uav team mission planning and execution. *AIAA Journal of Guidance, Control, and Dynamics*, 29(2):303–313, 2006.

[16] E. Johnson and S. Kannan. Adaptive Trajectory Control for Autonomous Helicopters. *AIAA Journal of Guidance, Control, and Dynamics*, 28(3):524–538, 2005.

[17] S. Park, J. Deyst, and J. P. How. A New Nonlinear Guidance Logic for Trajectory Tracking. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2005.

[18] P. R. Chandler and M. Pachter. Research Issues in Autonomous Control of Tactical UAVs. In *Proceedings of the IEEE American Control Conference*, pages 394–398, Washington, DC, June 1998.

[19] E. Frazzoli. *Robust Hybrid Control for Autonomous Vehicle Motion Planning.* PhD thesis, MIT, June 2001.

[20] J. C. Latombe. *Robot Motion Planning.* Kluwer Academic, 1991.

[21] S. M. LaValle. *Planning Algorithms.* Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.

[22] A. E. Bryson and Y. C. Ho. *Applied Optimal Control.* John Wiley & Sons Inc, 1979.

[23] D. Kirk. *Optimal Control Theory.* Dover, 1970.

[24] A. Rao and K. Clarke. Performance Optimization of a Maneuvering Re-Entry Vehicle Using a Legendre Pseudospectral Method. In *AIAA Atmospheric Flight Mechanics Conference*, number AIAA-2002-4885, Monterey, CA, Aug 2002.

[25] I. M. Ross and F. Fahroo. Pseudospectral Knotting Methods for Solving Nonsmooth Optimal Control Problems. *AIAA Journal of Guidance, Control, and Dynamics*, 27(3):397–405, May-June 2004.

[26] R. Franz, M. Milam, , and J. Hauser. Applied Receding Horizon Control of the Caltech Ducted Fan. In *Proceedings of the IEEE American Control Conference*, 2002.

[27] M. B. Milam, K. Mushambi, and R. M. Murray. A New Computational Approach to Real-Time Trajectory Generation for Constrained Mechanical Sys-

tems. In *Proceedings of the IEEE Conference on Decision and Control*, pages 845–851, Washington DC, 2000.

[28] E. Camacho and C. Bordons. *Model Predictive Control.* Springer-Verlag, 2004.

[29] J. M. Maciejowski. *Predictive Control with Constraints.* Prentice Hall, 2002.

[30] 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.

[31] J. A. Rossiter. *Model-Based Predictive Control: A Practical Approach.* CRC Press, 2003.

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

[33] W. B. Dunbar and R. Murray. Model Predictive Control of Coordinated Multi-vehicle Formations. In *Proceedings of the IEEE Conference on Decision and Control*, 2002.

[34] D.H. Shim, H. J. Kim, and S. Sastry. Decentralized Nonlinear Model Predictive Control of Multiple Flying Robots. In *Proceedings of the IEEE Conference on Decision and Control*, 2003.

[35] J. A. Primbs. The analysis of optimization based controllers. *Automatica*, 37 (6):933–938, 2001.

[36] James A. Primbs and Vesna Nevistic. A New Approach to Stability Analysis for Constrained Finite Receding Horizon Control Without End Constraints. *IEEE Transactions on Automatic Control*, 8(45):1507–1512, August 2000.

[37] K. Culligan. Online Trajectory Planning for UAVs Using Mixed Integer Linear Programming. Master's thesis, MIT, August 2006.

[38] T. Schouwenaars. *Safe Trajectory Planning of Autonomous Vehicles.* PhD thesis, MIT, December 2005.

[39] A. Bemporad and M. Morari. Robust Model Predictive Control: A Survey. *Lecture Notes in Control and Information Sciences*, 245:207–226, 1999.

[40] A. Casavola, M. Giannelli, and E. Mosca. Min-max predictive control strategies for input-saturated polytopic uncertain systems. *Automatica*, 36:125–133, 2000.

[41] L. Chisci, J. A. Rossiter, and G. Zappa. Systems with Persistent Disturbances: Predictive Control with Restricted Constraints. *Automatica*, 37(7):1019–1028, 2001.

[42] F. A. Cuzzolaa, J. C. Geromel, and M. Morari. An improved approach for constrained robust model predictive control. *Automatica*, 38(7):1183 1189, 2002.

[43] H. Fukushima and R.R. Bitmead. Robust Constrained Predictive Control using Comparison Model. *Automatica*, 41:97–106, 2005.

[44] E. C. Kerrigan and J. M. Maciejowski. Robust feasibility in model predictive control. In *Proceedings of the IEEE Conference on Decision and Control*, 2001.

[45] M. V. Kothare, V. Balakrishnan, and M. Morari. Robust Constrained Model Predictive Control using Linear Matrix Inequalities. *Automatica*, 32(10):1361–1379, 1996.

[46] J. Lofberg. *Minimax approaches to robust model predictive control.* PhD thesis, Linkoping University, 2003.

[47] P. O. M. Scokaert and D. Q. Mayne. Min-Max Feedback Model Predictive Control for Constrained Linear Systems. *IEEE Transactions on Automatic Control*, 8(43):1136–1142, August 1998.

[48] Z. Wan and M. V. Kothare. Robust Output Feedback Model Predictive Control Using Off-line Linear Matrix Inequalities. In *Proceedings of the IEEE American Control Conference*, June 2001.

[49] A. Abate and L. El Ghaoui. Robust Model Predictive Control through Adjustable Variables: an application to Path Planning. In *Proceedings of the IEEE Conference on Decision and Control*, December 2004.

[50] J. R. Gossner, B. Kouvaritakis, and J. A. Rossiter. Stable Generalized Predictive Control with Constraints and Bounded Disturbances. *Automatica*, 33(4): 551–568, 1996.

[51] H. Michalska and D. Q. Mayne. Robust Receding Horizon Control of Constrained Nonlinear Systems. *IEEE Transactions on Automatic Control*, 38(11): 1623–1633, November 1993.

[52] A. Richards. *Robust Constrained Model Predictive Control.* PhD thesis, MIT, November 2004.

[53] S. Butenko, R. Murphey, and P. Pardalos (Eds.). *Recent Developments in Cooperative Control and Optimization*, volume 3. Kluwer Academic Publishers, 2004.

[54] D. Grundel, R. Murphey, and P. Pardalos (Eds.). *Theory and Algorithms for Cooperative Systems*, volume 4 of *Series on Computers and Operations Research.* World Scientific Publishing Co., 2004.

[55] T. McLain, P. Chandler, S. Rasmussen, and M. Pachter. Cooperative Control of UAV Rendezvous. In *Proceedings of the IEEE American Control Conference*, pages 2309 – 2314, Arlington, VA, June 2001.

[56] R.O. Saber, W.B. Dunbar, and R.M. Murray. Cooperative control of multi-vehicle systems usiing cost graphs and optimization. In *Proceedings of the IEEE American Control Conference*, 2003.

[57] E. Camponogara, D. Jia, B. H. Krogh, and S. Talukdar. Distributed model predictive control. *IEEE Control Systems Magazine*, 2:44–52, 2002.

[58] M. G. Singh and A. Titli. *Systems: Decomposition, Optimisation and Control.* Pergamon Press, 1978.

[59] D. Jia and B. Krogh. Min-max feedback model predictive control for distributed control with communication. In *Proceedings of the IEEE American Control Conference*, pages 4507–45, 2002.

[60] S. L. Waslander, G. Inalhan, and C. J. Tomlin. Decentralized optimization via Nash bargaining. 2003.

[61] T. Keviczky, F. Borrelli, and G.J. Balas. A study on decentralized receding horizon control for decoupled systems. In *Proceedings of the IEEE American Control Conference*, 2004.

[62] T. Schouwenaars, J. 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*, August 2004.

[63] D. P. Bertsekas. Separable Dynamic Programming and Approximate Decomposition Methods. Report 2684, Laboratory for Information and Decision Systems, MIT, Feb 2006.

[64] M. Flint, M. Polycarpou, and E. Fernandez-Gaucherand. Cooperative path-planning for autonomous vehicles using dynamics programming. In *15th IFAC World Congress*, 2002.

[65] A. N. Venkat and J. B. Rawlings and S. J. Wright. Plant-wide optimal control with decentrlized MPC. Technical report, Department of Chemical and Biological Engineering, University of Wisconsin, 2004.

[66] Y. Kuwata and J. How. Three Dimensional Receding Horizon Control for UAVs. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference.* AIAA, August 2004.

[67] Y. Kuwata, A. Richards, T. Schouwenaars, and J. How. Robust Constrained Receding Horizon Control for Trajectory Planning. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, August 2005.

[68] Y. Kuwata and J. How. Stable Trajectory Design for Highly Constrained Environments using Receding Horizon Control. In *Proceedings of the IEEE American Control Conference*, pages 902–907. IEEE, June 2004.

[69] A. Richards and J. How. Model Predictive Control of Vehicle Maneuvers with Guaranteed Completion Time and Robust Feasibility. In *Proceedings of the IEEE American Control Conference*, Denver, CO, June 2003. IEEE.

[70] T. Schouwenaars, J. How, and E. Feron. Receding Horizon Path Planning with Implicit Safety Guarantees. In *Proceedings of the IEEE American Control Conference*, Boston, MA, July 2004. IEEE.

[71] Y. Kuwata, A. Richards, T. Schouwenaars, and J. How. Decentralized Robust Receding Horizon Control for Multi-vehicle Guidance. In *Proceedings of the IEEE American Control Conference*, pages 2047–2052, Minneapolis, MN, June 2006.

[72] Y. Kuwata and J. How. Decentralized Cooperative Trajectory Optimization for UAVs with Coupling Constraints. In *Proceedings of the IEEE Conference on Decision and Control*, pages 6820–6825, San Diego, CA, Dec 2006.

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

[74] Z. H. Mao, E. Feron, and K. Bilimoria. Stability and Performance of Intersecting Aircraft Flows Under Decentralized Conflict Avoidance Rules. *IEEE Transactions on Intelligent Transportation Systems*, 2(2):101–109, 2001.

[75] A. Richards, T. Schouwenaars, J. How, and E. Feron. Spacecraft Trajectory Planning With Collision and Plume Avoidance Using Mixed-Integer Linear Pro-

gramming. *AIAA Journal of Guidance, Control, and Dynamics*, 25(4):755–764, Aug 2002.

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

[77] 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.

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

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

[80] J. Bellingham, Y. Kuwata, and J. How. Stable Receding Horizon Trajectory Control for Complex Environments. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Austin, TX, Aug 2003. AIAA. (AIAA Paper 2003-5635).

[81] M. Zabarankin, S. Uryasev, and R. Murphey. Aircraft Routing under the Risk of Detection. *Naval Research Logistics*, 53:728–747, 2006.

[82] J. Canny and J. Reif. New Lower Bound Techniques for Robot Mmotion Planning Problems. In *28th Annual Symposium on IEEE Symposium on Foundations of Computer Science*, pages 49–60, Los Angeles, CA, October 1987.

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

[84] P. Narvaez, K.-Y. Siu, and H.-Y. Tzeng. New Dynamic Algorithms for Shortest Path Tree Computation. *IEEE/ACM Transactions on Networking*, 8(6):734–746, December 2000.

[85] A. Richards and J. P. 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.

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

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

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

[89] P. J. Goulart, E. C. Kerrigan, and D. Ralph. Efficient robust optimization for robust control with constraints. Technical report, Department of Engineering, University of Cambridge, May 2005. CUED/F-INFENG/TR.495.

[90] 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.

[91] H. Genceli and M. Nikolaou. Robust Stability Analysis of Constrained l_1-norm Model Predictive Control. *AIChE J.*, 39(12):19541965, 1993.

[92] Z. Q. Zheng and M. Morari. Robust Stability of Constrained Model Predictive Control. In *Proceedings of the IEEE American Control Conference*, page 379383, 1993.

[93] A.G. Richards and J.P. How. Robust Stable Model Predictive Control with Constraint Tightening. In *Proceedings of the IEEE American Control Conference*, 2006.

[94] A. Ben-Tal, S. Boyd, and A. Nemirovski. Extending Scope of Robust Optimization: Comprehensive Robust Counterparts of Uncertain Problems. *Mathematical Programming*, 107(1-2):63–89, 2006.

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

[96] A. Richards and J. How. Robust Model Predictive Control with Imperfect Information. In *Proceedings of the IEEE American Control Conference*, 2005.

[97] S.V. Rakovic, 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):pp.406–410, March 2005.

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

[99] M. Kvasnica, P. Grieder, M. Baotic, and F. J. Christophersen. *Multi-Parametric Toolbox (MPT).* ETH, June 2004.

[100] A. Bemporad. Reducing Conservativeness in Predictive Control of Constrained Systems with Disturbances. In *Proceedings of the IEEE Conference on Decision and Control*, pages pp. 1384–1391, Tampa, FL, 1998.

[101] J. Lofberg. Approximations of closed-loop minimax MPC. In *Proceedings of the IEEE Conference on Decision and Control*, pages 1438–1442, 2003.

[102] A. Richards and J. How. Decentralized Model Predictive Control of Cooperating UAVs. In *Proceedings of the IEEE Conference on Decision and Control*, December 2004.

[103] A. Richards and J. How. Implementation of robust decentralized model predictive control. In *AIAA GNC*, 2005.

[104] A. Richards, Y. Kuwata, and J. How. Experimental Demonstrations of Real-time MILP Control. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Austin, TX, Aug 2003.

[105] M. Valenti, B. Bethke, G. Fiore, and J. How. Indoor Multi-Vehicle Flight Testbed for Fault Detection, Isolation, and Recovery. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Keystone, CO, Aug 2006.

[106] Vicon MX Systems, July 2006. URL `http://www.vicon.com/products/viconmx.html`.

[107] Draganfly Innovations Inc. Draganfly V Ti Pro website, January 2006. URL `http://www.rctoys.com/draganflyer5tipro.php`.

[108] UAV SWARM Project Website, December 2006. URL `http://vertol.mit.edu/index.html`.

[109] I. Kolmanovsky and E. G. Gilbert. Maximal Output Admissible Sets for Discrete-Time Systems with Disturbance Inputs. In *Proceedings of the IEEE American Control Conference*, Seattle WA, 1995.

[110] A. Richards and J. How. Decentralized Algorithm for Robust Constrained Model Predictive Control. In *Proceedings of the IEEE American Control Conference*, Boston, MA, 2004. IEEE.

[111] Steven Skiena. *Implementing discrete mathematics: combinatorics and graph theory with Mathematica.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.

[112] W.B. Dunbar and R.M. Murray. Receding horizon control of multi-vehicle formation: A distributed implementation. In *Proceedings of the IEEE Conference on Decision and Control*, 2004.

[113] G. Inalhan, D. M. Stipanovic, and C. J. Tomlin. Decentralized Optimization, with Application to Multiple Aircraft Coordination. In *Proceedings of the IEEE Conference on Decision and Control*, December 2002.

[114] E. King, M. Alighanbari, Y. Kuwata, and J. How. Coordination and Control Experiments on a Multi-vehicle Testbed. In *Proceedings of the IEEE American Control Conference*, pages pp.5315–5320, Boston, MA, 2004. IEEE.

[115] A.N. Venkat, J.B. Rawlings, and S.J. Wright. Stability and optimality of distributed model predictive control. In *IEEE Conference on Decision and Control, and the European Control Conference*, 2005.

[116] U. Ozguner and W. R. Perkins. Optimal control of multilevel large-scale systems. *International Journal of Control*, 28(6):967–980, 1978.

[117] P. Heiskanen. Decentralized method for computing Pareto solutions in multiparty negotiations. *European journal of Operational Research*, 117:578–590, 1999.

[118] R.L. Raffard, C.J. Tomlin, and S.P. Boyd. Distributed optimization for cooperative agents: Application to formation flight. In *Proceedings of the IEEE Conference on Decision and Control*, 2004.

[119] D.P. Bertsekas and J.N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods.* Athena Scientific, 1997.

[120] D. P. Bertsekas. *Nonlinear Programming.* Athena Acientific, 1995.

[121] D. Q. Mayne, M. M. Seron, and S. V. Rakovic. Robust model predictive control of constrained linear systems with bounded disturbances. *Automatica*, 41(2): 219–224, February 2005.

[122] A. Casavola, E. Mosca, and M. Papini. Control under constraints: An application of the command governor approach to an inverted pendulum. *IEEE Transactions on Control Systems Technology*, 12(1):193–204, January 2004.

[123] N. Lynch. *Distributed Algorithms.* Morgan Kaufmann Publishers, San Mateo, CA, 1996.