Safe Trajectory Planning of Autonomous Vehicles

by

Tom Schouwenaars

Burgerlijk Elektrotechnisch Ingenieur
Katholieke Universiteit Leuven (2001)

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 2006

Author .....
Department of Aeronautics and Astronautics
November 18, 2005

Certified by .....
Eric Feron
Associate Professor of Aeronautics and Astronautics
Thesis Supervisor

Certified by .....
Jonathan P. How
Associate Professor of Aeronautics and Astronautics
Thesis Supervisor

Certified by .....
Munther Dahleh
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

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

Safe Trajectory Planning of Autonomous Vehicles

by

Tom Schouwenaars

Submitted to the Department of Aeronautics and Astronautics
on November 18, 2005, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

This thesis presents a novel framework for safe online trajectory planning of unmanned vehicles through partially unknown environments. The basic planning problem is formulated as a receding horizon optimization problem using mixed-integer linear programming (MILP) to incorporate kino-dynamic, obstacle avoidance and collision avoidance constraints. Agile vehicle dynamics are captured through a hybrid control architecture that combines several linear time-invariant modes with a discrete set of agile maneuvers. The latter are represented by affine transformations in the state space and can be described using a limited number of parameters. We specialize the approach to the case of a small-scale helicopter flying through an urban environment.

Next, we introduce the concept of terminal feasible invariant sets in which a vehicle can remain for an indefinite period of time without colliding with obstacles or other vehicles. These sets are formulated as affine constraints on the last state of the planning horizon and as such are computed online. They guarantee feasibility of the receding horizon optimization at future time steps by providing an a priori known backup plan that is dynamically feasible and obstacle-free. Vehicle safety is ensured by maintaining a feasible return trajectory at each receding horizon iteration. The feasibility and safety constraints are essential when the vehicle is maneuvering through environments that are only partially characterized and further explored online. Such a scenario was tested on an unmanned Boeing aircraft using scalable loiter circles as feasible invariant sets.

The terminal feasible invariant set concept forms the basis for the construction of a provably safe distributed planning algorithm for multiple vehicles. Each vehicle then only computes its own trajectory while accounting for the latest plans and invariant sets of the other vehicles in its vicinity, i.e., of those whose reachable sets intersect with that of the planning vehicle. Conflicts are solved in real-time in a sequential fashion that maintains feasibility for all vehicles over all future receding horizon iterations. The algorithm is applied to the free flight paradigm in air traffic control and to a multi-helicopter relay network aimed at maintaining wireless line of sight communication in a cluttered environment.

Thesis Supervisor: Eric Feron
Title: Associate Professor of Aeronautics and Astronautics

Thesis Supervisor: Jonathan P. How
Title: Associate Professor of Aeronautics and Astronautics

Thesis Supervisor: Munther Dahleh
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

This dissertation is the result of a fruitful collaboration with my advisor Prof. Eric Feron and co-advisor Prof. Jonathan How. I am extremely grateful for their guidance over the last four years and for creating an environment in which I enjoyed significant academic and organizational freedom. I am indebted to Eric's unconditional support, both on a moral and financial level (including sponsoring for many trips), and thank him for the continued trust he placed in me. His energy, openness, and out of the box ideas have never stopped impressing me. He made the lab a fun place to work. Jon was in a way my process and quality control manager. I thank him for the many detailed discussions on research problems and patient revisions of (usually last-minute) paper manuscripts. His dedication to research, publishing and teaching are exemplary. Finally, I would like to thank my third committee member, Prof. Munther Dahleh, for his valuable input during the last semesters.

The long days (and nights) in LIDS would not have been the same without the social contacts with my colleague students, many of whom became close friends. I have vivid memories of the daily conversations and sharing of frustrations (in Flemish) with Jan De Mot, the lunches and cookie follow-ups with Masha Ishutkina, the espresso breaks and laughs with Julius Kusuma, the Boeing visits and related 3 AM food trips to La Verde's with Mario Valenti, the humor and cycling expertise of Chris Dever, the discussions on intellectual property with Gregory Mark, the introduction to Iranian culture by Mardavij Roozbehani, the research and conference trips with Bernard Mettler, the late night exchange of politics opinions with Selcuk Bayraktar, and the discussions on academia vs. industry with Vishwesh Kulkarni. Over the years, the friendly and cooperative atmosphere in the lab was further completed by Animesh Chakravarthy, Han-Lim Choi, Emily Craparo, David Dugail, Vladislav Gavrilets, Sommer Gentry, John Harper, Farmey Joseph, Georgios Kotsalis, Jerome Le Ny, Rodin Lyasoff, Ioannis Martinos, Nuno Martins, Mitra Osqui, Navid Sabagghi, Keith Santarelli, Danielle Taraf, Olivier Toupet, Glenn Tournier, and Ji Hyun Yang, all of whom I frequently interacted with in the office or corridor. The administrative staff made all processes run smoothly. A special word of thanks goes to Lauren Clark, Angela Copyak, Kathryn Fisher, Lisa Gaumond, Doris Inslee, and Margaret Yoon, and to Marie Stuppard for providing flexibility with departmental deadlines on several occasions.

During the course of my studies, I had the opportunity to work with various non-MIT organizations, which gave me insight into how engineering is practised outside of the academic world. I would like to thank Dr. Mark Tischler for hosting me at Nasa Ames and Boeing Phantom Works in St. Louis for the cooperation on the Software Enabled Control Program. Last summer, I worked intensely with Dr. Andrew Stubbs and Dr. James Paduano from Nascent Technology Corporation. I am also grateful to Prof. Emile Boulpaep, the Belgian American Educational Foundation and the Francqui Foundation for providing the financial support to start my studies at MIT.

Despite the many hours spent in the lab and working on problem sets at home, there *was* a life beside these duties, for which I could count on a large group of friends. The MIT European Club gave me the chance to actively participate in the foreign student scene in the Boston area. The rich social interactions added to the already incredible mix of cultures at MIT and resulted in many friendships for life. I am also grateful to Michael Garcia-Webb, my roommate during the last years, for the many conversations in the evenings and for coping with my shifted night schedule. I also have great memories of the weekend parties with our Sidney-Pacific friends, at least of those I did not have to bail out because of some upcoming deadline. On my short trips home, family and friends were always available for

having a good time, and Klaartje Genbrugge's regular emails and the moral support of my soulmate Carolina Diaz-Quijano have kept me sane during the last months.

Finally, I am grateful to my parents for supporting me in my choices and making me believe in myself. To them I dedicate this thesis.

Contents

1 Introduction **17**
- 1.1 Autonomous Trajectory Planning . . . 17
  - 1.1.1 Unmanned Vehicles . . . 17
  - 1.1.2 Trajectory Planning Problem . . . 18
  - 1.1.3 Guidance System Hierarchy . . . 19
- 1.2 Literature Overview . . . 20
  - 1.2.1 Non-MPC Trajectory Planning Methods . . . 20
  - 1.2.2 MPC-based Trajectory Planning . . . 22
- 1.3 Statement of Contributions . . . 23
- 1.4 Thesis Outline . . . 25

2 Receding Horizon Trajectory Planning **27**
- 2.1 Introduction . . . 27
- 2.2 Problem Formulation . . . 28
  - 2.2.1 Problem Setup . . . 28
  - 2.2.2 Receding Horizon Planning . . . 29
  - 2.2.3 Optimization Problem . . . 30
- 2.3 MILP Formulation . . . 31
  - 2.3.1 Mixed Integer Linear Programming . . . 32
  - 2.3.2 Vehicle Dynamics . . . 32
  - 2.3.3 Obstacle Avoidance . . . 34
  - 2.3.4 Collision Avoidance . . . 37
- 2.4 Example Scenarios . . . 37
  - 2.4.1 Example 1: UAV in 2D . . . 37
  - 2.4.2 Example 2: Multiple UAVs in 2D . . . 38
  - 2.4.3 Example 3: Helicopter in 3D . . . 39
- 2.5 Conclusion . . . 40

3 Hybrid Model for Agile Vehicles **43**
- 3.1 Introduction . . . 43
- 3.2 Hybrid Control Architecture for Guidance . . . 44
  - 3.2.1 Automatic Control of Agile Vehicles . . . 44
  - 3.2.2 Velocity Control System . . . 45
  - 3.2.3 Maneuver Scheduler . . . 47
  - 3.2.4 LTI-Maneuver Automaton . . . 48
- 3.3 Trajectory Optimization with the LTI-MA . . . 48
  - 3.3.1 Sequential decision process . . . 48

3.3.2 Trajectory Optimization Using MILP . . . 49
3.3.3 Planning Strategies . . . 51
3.4 Small-Scale Helicopter Example . . . 53
3.4.1 Helicopter LTI Modes . . . 53
3.4.2 Helicopter Maneuvers . . . 58
3.4.3 Obstacle Avoidance . . . 60
3.4.4 Receding Horizon Formulation . . . 62
3.5 Results . . . 63
3.5.1 Helicopter and Problem Parameters . . . 63
3.5.2 Scenarios . . . 64
3.5.3 Hardware in the Loop Experiments . . . 68
3.6 Practical Considerations . . . 68
3.7 Conclusion . . . 69

4 Trajectory Planning with Feasibility and Safety Guarantees 71
4.1 Introduction . . . 71
4.2 Unsafe Scenarios . . . 72
4.3 Feasibility Constraints . . . 73
4.3.1 Feasible Invariant Sets . . . 73
4.3.2 Feasible Receding Horizon Planning Problem . . . 75
4.4 Feasible Invariant Sets as Affine Transformations . . . 76
4.4.1 Terminal Feasible Invariant Set Modeling . . . 76
4.4.2 Sampling Points Requirements . . . 77
4.4.3 MILP Formulation . . . 79
4.4.4 Examples . . . 80
4.5 Safety Constraints . . . 82
4.5.1 Safety Definitions . . . 82
4.5.2 Safe Feasible Receding Horizon Planning Problem . . . 85
4.5.3 Backtrack Pattern . . . 88
4.5.4 Safe Receding Horizon Planning without Feasibility Guarantees . . . 89
4.6 Conclusion . . . 91

5 Safe Distributed Trajectory Planning for Multiple Vehicles 93
5.1 Introduction . . . 93
5.2 Problem Formulation . . . 95
5.2.1 Receding Horizon Planning . . . 95
5.2.2 Safety Principle . . . 96
5.2.3 Conflict Description . . . 96
5.2.4 Communication Requirements . . . 97
5.3 Safe Trajectory Planning Algorithm . . . 100
5.3.1 Algorithm . . . 100
5.3.2 Remarks . . . 101
5.4 Implementation Using MILP . . . 102
5.4.1 Aircraft Model . . . 103
5.4.2 Loiter Circles . . . 104
5.4.3 Avoidance Constraints . . . 104
5.4.4 Cost Function . . . 106
5.5 Results . . . 106

5.6 Conclusion . . . 108

6 Multi-Vehicle Path Planning for Non-Line of Sight Communication 111
6.1 Introduction . . . 111
6.2 Problem Formulation . . . 113
6.2.1 Problem Setup . . . 113
6.2.2 Centralized Receding Horizon Planning . . . 114
6.2.3 Connectivity Constraints . . . 115
6.3 Implementation . . . 116
6.3.1 Helicopter Test-Bed . . . 116
6.3.2 Mission Scenario . . . 117
6.3.3 Trajectory Planning Software . . . 118
6.4 Results . . . 119
6.4.1 Planning Parameters . . . 119
6.4.2 Simulation and Flight Test Results . . . 120
6.5 Distributed Planning Strategy . . . 123
6.5.1 Distributed Cooperative Algorithm . . . 123
6.5.2 Results . . . 125
6.6 Conclusion . . . 126

7 Implementation of MILP-based UAV Guidance for Human/UAV Team 129
7.1 Introduction . . . 129
7.2 Experiment and Technology Overview . . . 130
7.2.1 Mission Scenario . . . 130
7.2.2 Technology Development . . . 133
7.3 Natural Language Parsing and Interfacing . . . 133
7.4 Task Scheduling and Communications Interfacing . . . 135
7.5 MILP-based Trajectory Generation Module . . . 136
7.5.1 MILP Formulation . . . 136
7.5.2 Implementation . . . 139
7.6 Simulation and Flight Test Results . . . 141
7.6.1 Simulation Results . . . 142
7.6.2 Flight-Test Results . . . 144
7.7 Conclusion . . . 148

8 Conclusion 149
8.1 Summary . . . 149
8.2 Future Work . . . 150

List of Figures

1-1 Boeing's Unmanned Combat Aerial Vehicle (UCAV) 18
1-2 Hierarchical decomposition of a guidance system into three decision and control layers 19

2-1 Approximation of minimum and maximum velocity bounds by polygons. . . 33
2-2 Approximation of an arbitrary 2D obstacle by its convex hull and with convex polygons filling up the concavities. The vehicle is allowed to be inside the red areas, but must stay outside the grey area. 35
2-3 Example 1: The aircraft is initially in the origin, flying east at 4 m/s, and has to maneuver to position $(70, 57)$ m. The goal is reached after 28 s. . . . 38
2-4 Example 2: The aircraft are initially positioned along a circle and need to fly to the opposite side. A conflict in the middle is avoided thanks to the collision avoidance constraints. 39
2-5 Example 3: The helicopter is initially hovering in position $(30, -60, 8)$ m, and has to maneuver to hover in position $(125, 60, 10)$ m. The goal is reached after 14 s. 40
2-6 Two-dimensional projection of Example 3 41

3-1 Hybrid control architecture: the helicopter can be controlled through a velocity control system, or through a maneuver scheduler that allows the implementation of agile maneuvers taken from a library. 46
3-2 Abstract representation of the control architecture. The linear controllers allow continuous motion, the maneuvers are discrete transitions. 46
3-3 Graph representation of the LTI-maneuver automaton. The vehicle is either in an LTI mode, or executing a finite duration maneuver. 48
3-4 Example of operating regions of the velocity control mode, shown in function of the body axis forward (u) and lateral velocity (v). 54
3-5 Two circle approximation of the elliptic constraint on forward and lateral acceleration. 56
3-6 Polygonal approximation of minimum and maximum velocity bounds delimiting the various LTI-modes. 57
3-7 Helicopter trajectories for sample maneuvers. 59
3-8 Change in helicopter position and orientation resulting from a maneuver (shown in dashed line), as observed from the body-fixed frame 60
3-9 Scenario 1: The helicopter is initially in $(0, 0)$ flying at 12 m/s and needs to reverse direction towards $(-100, 0)$ as quickly as possible. It decides to execute the split-S. 65

3-10 Scenario 2: The helicopter is initially in $(0, 0)$ flying at 6 m/s and needs to reverse direction towards $(-100, 0)$ as quickly as possible. It decides to make a U-turn. 65
3-11 Scenario 3: The helicopter is initially in $(0, 0)$ flying at 12 m/s and needs to reverse direction towards $(-100, 0)$ as quickly as possible. To avoid the obstacle in the middle, it decides to change heading before executing the split-S. 66
3-12 Scenario 4: The helicopter is initially in $(0, 0)$ flying at 6 m/s and needs to reverse direction towards $(-100, 0)$ as quickly as possible. To avoid the obstacle in the middle, it decides to make wider a U-turn. 66
3-13 Scenario 5: The helicopter is initially in $(0, 0)$ flying at 12 m/s and needs to reverse direction towards $(-100, 0)$ as quickly as possible. Because of the narrower corridor, the vehicle first turns northeast before executing the split-S. 67
3-14 Scenario 6: The helicopter is initially in $(0, 0)$ flying at 12 m/s and needs to reverse direction towards $(-100, 0)$ as quickly as possible. The only feasible option is to slow down and make a turn on the spot. 67
3-15 Hardware in the loop experiments. The helicopter is flying between two obstacles, initially heading east at 10 m/s. Figure (a) shows the helicopter reversing direction by slowing down till hover and turning on the spot. In Figure (b), it performs a hammerhead. The different actions depend on the width of the corridor. 68

4-1 The aircraft is initially in the origin, flying east at 4 m/s, and has to maneuver to position $(70, 57)$ m. After 17 s, the MILP becomes infeasible, corresponding to the aircraft colliding with the obstacles. 73
4-2 The aircraft are initially positioned along a circle and need to fly to the opposite side. Near the origin, the MILP becomes infeasible: the aircraft have approached each other too closely and an intersection of their avoidance zones cannot be avoided. 74
4-3 Safe trajectory ending in either a right or left turning loiter circle. 77
4-4 Situation where undersampling of the loiter circle leads to a safety violation. Although the obstacle avoidance constraints for the sample points are satisfied, the circle intersects the obstacle. 78
4-5 Situation where the loiter circle cuts the corner of an obstacle. This situation can be avoided by enlarging the obstacles with a safety boundary d_{safe}. . . . 78
4-6 Receding horizon trajectory with loiter constraints. Because of the loiter constraints, the UAV avoids the concavity and chooses an alternative route to reach the goal. 80
4-7 Sequence of intermediate receding horizon trajectories ending in loiter circles. 80
4-8 Receding horizon trajectory with loiter constraints. Although the UAV enters the concavity, it does not crash and the trajectory ends in a loiter. 81
4-9 Sequence of intermediate receding horizon trajectories for a wider concavity. Although the UAV enters the concavity, it does not crash and the trajectory ends in a loiter. 81
4-10 Feasible receding horizon trajectory with loiter constraints. 82
4-11 Sequence of intermediate receding horizon trajectories with loiter constraints consisting of 24 sample points. 82

4-12 Scenario in which the terminal feasible invariant set constraints cannot prevent the UAV from getting trapped inside a concavity. Although feasibility is maintained, actual vehicle safety is not. 83
4-13 Intermediate receding horizon trajectories ending in loiter circles for the scenario in which the UAV gets trapped inside a concavity. 84
4-14 Safe receding horizon trajectory of T time steps ending in a terminal feasible invariant set $\mathcal{S}(t+T|t)$ and with backtrack pattern $\mathcal{B}(t+T|t)$ to the return trajectory $\mathcal{T}(t)$. 85
4-15 Intermediate receding horizon trajectories with backtrack pattern. The vehicle now avoids entering and getting trapped in the concavity because the loiter circle with additional backtrack pattern do not fit inside of it. 88
4-16 Intermediate receding horizon trajectories with backtrack patterns that are not geometrically constrained and can join the return trajectory in any one state of a given subset. 89
4-17 Safe receding horizon trajectory without feasibility guarantees. The red triangles indicate positions where the problem became infeasible (but safety was maintained). 90
4-18 Intermediate receding horizon trajectories with safety patterns that do not contain a terminal invariant set. 90

5-1 Simulation results for distributed conflict resolution of 2 and 4 aircraft for 30 time steps of 5 s each. 107
5-2 Computation times at each iteration for the 2 aircraft scenario of Figure 5-1a. 108
5-3 Computation times at each iteration for the 4 aircraft scenario of Figure 5-1b. 108
5-4 Safe trajectories and corresponding loiter boxes for the 4 aircraft scenario during the first 9 time steps (corresponding to 45 s). 109
5-5 Safe trajectories of 8 aircraft during 40 time steps of 5 s each. 109
5-6 Safe trajectories of 10 aircraft during 65 time steps of 5 s each. 110

6-1 MIT's autonomous X-Cell helicopter, equipped with avionics box. 116
6-2 Satellite image of the test region. The rectangle represents the take-off area and ground station, the circle is the target location. Source: Google Earth . 118
6-3 Nominal trajectories of the Burlington scenario computed in Matlab using receding horizon planning. The circles represent the mission helicopter, the triangles show the relay vehicle. The obstacles in the center are buildings; the large no-fly zone at the Western and Southern edge represents forests (enlarged with a safety boundary). 121
6-4 Hardware-in-the-loop simulations of the trajectories computed online in real-time. The circles represent the mission helicopter; the triangles show the relay vehicle. 121
6-5 Actual flight test data of the trajectories computed online in real-time. The circles represent the mission helicopter; the triangles show the relay vehicle. 122
6-6 Simulation result for distributed Burlington scenario with extra obstacle and second relay vehicle. 126
6-7 Simulation result for distributed scenario in which the mission helicopter must fly over the building (MIT's Simmons Hall) and the relay has to gain altitude. 127

7-1 SEC Capstone Demonstration test vehicles 131
7-2 Overview of the MIT flight experiment 132
7-3 Block diagram of the MIT SEC Capstone Demonstration system. "FCS" stands for Flight Control System, "GIB" for Guy-in-Back (i.e., the rear-seat operator). 134
7-4 Sample scenario map for the MIT SEC Capstone Demonstration. The flight area is approximately 40 mi long along the northern boundary. 141
7-5 SEC demonstration Simulation-In-the-Loop (SIL) laboratory setup 142
7-6 SIL Test 1 - Initialization of the SEC demonstration: the UAV (in light) enters a loiter pattern. 143
7-7 SIL Test 2 - Pop-up obstacle test: the UAV safely avoids both pop-up threats. 144
7-8 SIL Test 3 - Simulated flight with two consecutive search missions. 145
7-9 Flight experiment system level diagram 145
7-10 SEC Flight-Test: UAV/T-33 (in light) with simulated F-15 (in dark) 146
7-11 SEC Flight-Test: UAV/T-33 (in light) and actual F-15 (in dark) providing natural language commands. 147

List of Tables

3.1 Description of sample maneuvers that could be implemented on a rotorcraft-type vehicle. 58
3.2 Parameters of LTI modes 63
3.3 Parameters of pre-programmed maneuvers 64

6.1 Parameters used in the MILP trajectory optimization 119

Chapter 1

Introduction

1.1 Autonomous Trajectory Planning

1.1.1 Unmanned Vehicles

In recent years, autonomous vehicles have become increasingly important assets in various civilian and military operations. A wide variety of robotic vehicles is currently in use or being developed, ranging from unmanned fixed-wing aircraft, helicopters, blimps and hovercraft to ground and planetary rovers, earth-orbiting spacecraft and deep-space probes. Although the level of autonomy differs among the types of vehicles and the missions they are used for, many such systems require no or minor human control from a base or ground station. The primary reason for deploying autonomous vehicles is often a reduction in cost or elimination of human risk associated with a particular mission: unmanned systems do not require operator safety and life support systems, and can therefore be made smaller and cheaper than their manned counterparts. Furthermore, autonomous vehicles enable operations in remote or harsh environments and often possess the capability to operate continuously or complete missions that are of longer duration than ones manned systems are capable of.

In this dissertation, we are mainly interested in autonomous rotorcraft and fixed-wing aircraft, or so-called unmanned aerial vehicles (UAVs). One such vehicle, Boeing's UCAV, is pictured in Figure 1-1. Although their development has primarily been motivated by military needs [101], such as unmanned combat missions and reconnaissance and surveillance operations, there are many civilian applications of interest. They include search and rescue operations, surveillance of natural disaster sites such as (possibly remote) areas hit by an earthquake or flood, and inspection of environments that are typically inaccessible to humans, such as active volcanic craters or sites where high radioactive radiation is present. Other applications are urban surveillance, traffic monitoring, weather observation, freight services, and the creation of spectacular camera shots in the movie and advertising industry. Some of these might require cooperative behavior between multiple vehicles. For example, in an urban environment where wireless line of sight communication with a ground station is obstructed by buildings, a team of multiple vehicles can act as an autonomous relay network. Finally, autonomy technologies developed for UAVs can contribute to the improvement or automation of the commercial air traffic management system.

Figure 1-1: Boeing's Unmanned Combat Aerial Vehicle (UCAV)

1.1.2 Trajectory Planning Problem

Most UAVs that are currently operational, however, are flown remotely by a human pilot or track predetermined waypoint plans. The operating costs can be further decreased and the flexibility required in handling volatile or unexpected situations significantly improved by increasing the level of autonomy. An essential part of that vehicle autonomy consists of a trajectory planning and guidance system that enables it to safely maneuver through a particular environment. This environment may contain obstacles and zones that the vehicle is not allowed to enter and may not be fully characterized at the start of a mission. Obstacles may be detected as the vehicle moves through the environment or their location may change over time. A special case of non-stationary obstacles are other autonomous agents with which collisions should be avoided. The vehicle should thus have the capability to compute or update its path in real-time, i.e., as the mission unfolds, thereby accounting for its dynamic and kinematic properties.

This thesis tackles various problems associated with online trajectory planning of UAVs through cluttered environments and as such contributes to the desired increase in vehicle autonomy. A precise mathematical definition of the trajectory planning problem will be given in the next chapter, but for now the following less formal description suffices:

Trajectory Planning Problem: *Given the present state of a single vehicle or team of vehicles and a map of the environment, compute a trajectory towards a desired goal state or configuration in real-time that optimizes a certain objective function while respecting the kino-dynamic properties of the vehicle(s) and avoiding obstacles and collisions.*

In what follows, we will also refer to trajectory planning as trajectory optimization, path planning, or motion planning.

The specific characteristics of various vehicle types pose different challenges to the trajectory optimization. For example, a ground vehicle and helicopter have the ability to stop and go backwards, whereas a fixed-wing aircraft has to maintain a minimum velocity. Some rovers and helicopters can make quick turns on the spot, but have a slower turn rate when moving forward. Reconfiguring satellite clusters or spacecraft maneuvering around the International Space Station have the additional requirement to avoid plume impingement on solar panels when firing their thrusters. The framework presented in this dissertation will be able to accommodate such vehicle-specific constraints.

Although the trajectory planning methodology developed in this thesis can be applied to many autonomous vehicle types, we will focus on applications and scenarios for aerial vehicles. Among the various types of UAVs, miniature helicopters form a special class and

are particularly interesting for operations in cluttered environments. Besides their ability to hover at a fixed location, fly at low speeds, and turn on the spot, they can exhibit very agile behavior enabling quick obstacle avoidance in changing environments and fast nap of-the-earth flight. Taking advantage of these capabilities in an automated fashion is one of the problems tackled in this thesis.

1.1.3 Guidance System Hierarchy

To situate the trajectory planning problem more precisely, we consider a hierarchical decomposition of an autonomous guidance system into three decision and control layers [89]. These can be viewed as levels of abstraction encapsulating the physical capabilities of the vehicle and enabling an operator to interface with the vehicle at the mission assignment level. In this hierarchy, shown in Figure 1-2, the lowest level consists of the physical control layer. It stabilizes the vehicle dynamics and augments them with either an autopilot acting as a waypoint follower or with a velocity control system tracking velocity and turn rate commands. The waypoints or velocity commands are the output of the intermediate level, which is the trajectory or motion planning layer. This layer is responsible for guiding the vehicle from its present state to its desired location while optimizing a certain cost function, such as minimizing time, fuel or visibility, or maximizing the area explored in a search operation. The obstacle and collision avoidance requirements, together with a closed-loop model of the dynamics resulting from the lower control layer, thereby enter as constraints in the trajectory optimization problem.

The third layer and highest level of abstraction is the overall mission planner. It lays out

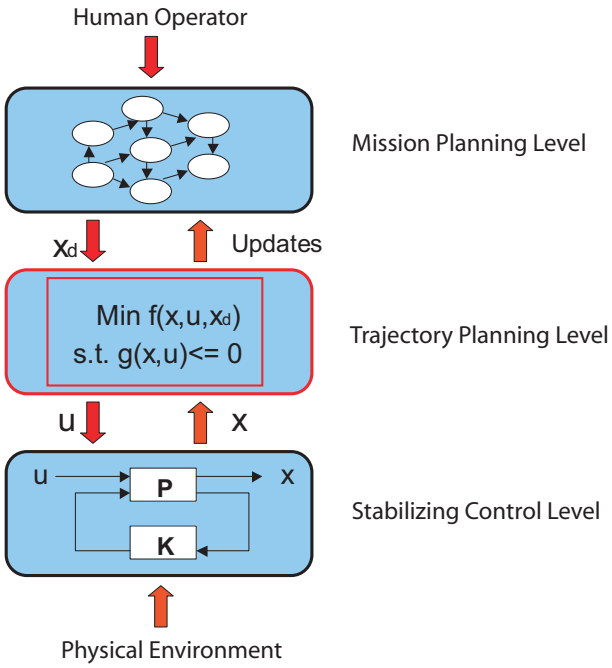


Figure 1-2: Hierarchical decomposition of a guidance system into three decision and control layers

a sequence of task waypoints that accomplish the particular mission and act as subsequent goal locations at the intermediate trajectory planning level. Examples of such higher level missions are the UAV applications given above. In case multiple vehicles are involved, this layer is also responsible for the individual task assignment [130, 6, 113, 78]. To remain computationally tractable, simplified descriptions of the environment and the vehicle's physical capabilities resulting from the lower levels are typically used when generating the mission plan [27, 28, 99]. However, the various models used in this hierarchy should be consistent, *i.e.* the behaviors commanded by the higher decision layers should be executable by the lower layers [104, 105].

We can now say more specifically that this dissertation develops the intermediate trajectory design level assuming that both the higher mission planning and lower physical control layers are in place.

1.2 Literature Overview

Motion planning has been an important research topic in the field of robotics and artificial intelligence for several decades. The earliest methods tackled trajectory planning for holonomic systems operating in an environment without obstacles and were based on optimal control [1, 24] and nonlinear programming techniques [16]. A survey of such numerical methods can be found in [20]. In this dissertation, however, we are interested in tackling the motion planning problem in the presence of obstacles and other vehicles. Many methods have been developed, and it is not our ambition to give a thorough survey of all of these. Instead, we will broadly distinguish between recent techniques based on model predictive control (MPC) and earlier methods, many of which were developed in the artificial intelligence community and some of which yielded fundamental complexity results [110, 49, 111, 26].

1.2.1 Non-MPC Trajectory Planning Methods

The classic monograph by Latombe [72] presents many of the basic concepts and complexity results related to robot motion planning. A more recent book by LaValle [74] also covers the latest methods in the field. Other survey works include the papers by Schwartz and Sharir [132, 133], Hwang and Narendra [54], and Latombe [73]. An overview of specific methods for spacecraft formation flying and reconfiguration can be found in the work of Sharf *et al.* [123, 124]. In addition, Frazzoli [35] gives a classification of the various kinodynamic motion planning problems in free or obstacle environments in terms of nonlinear control problems.

Among the basic concepts in motion planning is the notion of the configuration space, in which the size of the vehicle is added to the obstacles [80, 52]. As such, the vehicle itself is reduced to a point mass. In case the orientation of the vehicle is of importance to being able to pass an obstacle, e.g., in the so-called piano mover's problem [110, 131, 77], an extra dimension is introduced in which the size of the obstacle changes with the vehicle's orientation. The configuration space is generally used in the various algorithmic motion planning methods.

The first methods developed in the artificial intelligence community were based on dynamic programming algorithms [17, 138] for graph searching formulations of the motion planning problem. Among those are cell decomposition or surface covering methods in which the configuration space is partitioned into a finite number of regions and the motion

planning problem is reduced to finding a sequence of neighboring cells [80, 81]. Other graph searching methods are so-called roadmap techniques where a network of feasible paths is constructed among which a sequence of connecting paths between the initial and final configuration is selected. Visibility graphs [82] and Voronoi diagrams [30] are examples of such roadmap constructions.

Most of these early methods, however, use simplified kinematic models of the vehicles or robots, which may lead to conservative results. For example, the full dynamic capabilities may not be exploited, or a safety margin may have to be included that accounts for uncertainties in the actual motion when the kinematic reference trajectory is tracked by a lower level control law. Although they were initially used with kinematic models as well [65, 151, 3], artificial potential field methods do allow to account for the dynamics of the vehicle [92, 11, 119]. In these methods, obstacles and other vehicles are modeled as repelling forces in a potential field by being embedded as peaks in a potential function. Using numerical optimization techniques, the gradient of the latter is then used to steer the vehicle towards the global minimum of the function corresponding to the desired state.

Although potential function methods can produce smooth, dynamically feasible trajectories, they have several disadvantages. A first problem is that the vehicle might get trapped in a local minimum. A second issue arises from the fact that obstacles are not modeled as hard constraints. In addition, obstacles typically have to be modeled as continuous and differentiable functions, leading to an imprecise description of the obstacle's shape and dimensions. Especially when the vehicle has to maneuver through tight environments, these soft constraints cannot provide hard avoidance guarantees.

Alternative methods that produce dynamically feasible trajectories *with* hard avoidance guarantees are based on so-called rapidly-exploring random trees (RRTs) [75, 60]. These trees consist of feasible trajectories that are built online by extending branches towards randomly generated target states. The RRT methods form a subset of the larger class of stochastic optimization [88] and randomized motion planning algorithms [61, 2], the first of which were probabilistic roadmap (PRM) planners [62, 63, 51, 21]. PRM algorithms combine an offline construction of a roadmap with a randomized online selection of an appropriate path from the roadmap. Unlike RRT methods, however, these algorithms cannot be applied in rapidly changing environments due to the offline construction of the roadmap.

Randomized algorithms were introduced to circumvent the intrinsic exponential complexity of the motion planning problem. Reif [110] indeed showed that the problem of finding a path for a robot consisting of several polyhedral parts through an environment with polyhedral obstacles is PSPACE-hard. Schwartz and Sharir [131] constructed an algorithm for non-polyhedral obstacles whose time complexity is twice exponential in the dimension n of the configuration space and polynomial in the number and degree of polynomial constraints describing the obstacles. Canny [26] found a more efficient algorithm that is single exponential in n. Fundamental complexity results for problems involving multiple robots were derived by Hopcroft *et al.* [49] and Reif and Sharir [111]. A topological approach to establish the complexity of nonholonomic motion planning was carried out by Jean [58].

To further reduce the complexity associated with including the detailed vehicle dynamics in the problem formulation, methods using motion primitives were introduced. Marigo and Bicchi [87] used control quanta, whereas Frazzoli [35, 36] proposed the concept of a maneuver automaton. The latter consists of a discrete set of trim conditions and transitions between these trims that are called maneuvers. Using dynamic programming, a value function is computed offline that gives the optimal time between a particular vehicle state and a desired state, whereby the trajectories consist of a sequence of trims and maneuvers. The value

function is used in an online control policy and obstacle avoidance is obtained using rapidly-exploring random trees. In this dissertation, we will generalize this automaton framework to an approach that includes continuous trim modes and permits to include obstacle avoidance directly in the optimization.

Other recent methods that account for precise nonlinear dynamics are based on iterative methods using splines [95, 79] or nonlinear programming [53, 108]. To be applicable in real-time, however, they must be combined with a receding horizon planning strategy as is discussed next.

1.2.2 MPC-based Trajectory Planning

The trajectory planning methodology and concepts that are developed in this thesis belong to the class of model predictive control-based motion planning. Similarly to the randomized and motion primitive approaches described above, these algorithms aim at reducing the complexity of the planning problem and do so by repeatedly solving online a constrained optimization problem over a finite planning horizon. At each iteration, a segment of the total path is computed using on a dynamic model of the vehicle that predicts its future behavior. More specifically, a sequence of control inputs and resulting vehicle states is generated that meet the kino-dynamic and environment constraints and that optimize some performance objective. Only a subset of these inputs is actually implemented, however, and the optimization is repeated as the vehicle maneuvers and new measurements of the vehicle states and the environment are obtained. As such, this approach is especially useful when the environment is explored online.

Model predictive control (MPC) or receding horizon control (RHC) originated in the process industry a few decades ago, and has since received wide attention in the broader field of control theory and other applications. The benefits of the approach is that it allows to naturally handle multivariable sytems, can systematically take actuator limitations into account, and allows a system to operate closer to its constraints, therefore often resulting in better performance [85]. The survey papers by Garcia *et al.* [38], Morari and Li [98], Mayne *et al.* [91, 90], and Rawlings [109] offer a good introduction to the field.

Because of this flexibility, in recent years, receding horizon control has been introduced to the problem of trajectory planning as well. Initially, most work only tackled problems in obstacle-free environments [48, 57] or was limited to tracking predetermined trajectories around obstacles [139]. However, Bemporad *et al.* [11] have used potential functions to model obstacles and Dunbar *et al.* [32] have used splines to produce collision-free trajectories.

In [125, 127] an alternative receding horizon approach based on *mixed-integer linear programming* (MILP) was introduced. MILP is a powerful optimization framework that allows inclusion of integer variables and discrete logic in a continuous linear optimization problem. These variables can be used to model logical constraints such as obstacle and collision avoidance rules, while the dynamic and kinematic properties of the vehicle are formulated as continuous constraints. An overview of applications and algorithms to solve MILP problems can be found in [33]. It is still an active research topic in the field of Operations Research [144, 152], and the state of the art in cutting plane methods, branch-and-bound algorithms, integral basis methods, and approximation algorithms is covered in the recent book by Bertsimas and Weismantel [19].

The main advantage of using MILP for trajectory planning is that it can systematically handle *hard* obstacle and collision avoidance constraints and allows to include other

decision features such as task assignment into the optimization problem. Moreover, the algorithms are *complete*, i.e., they give a feasible solution if one exists. Although developed independently in [127], the MILP-based trajectory planning approach is a special case of the broader class of control for mixed logical dynamical (MLD) systems, developed by Bemporad and Morari in [12]. They provide a general method for formulating hybrid system dynamics in a mathematical framework using a combination of real and binary variables. Explicit solutions for piecewise linear optimal controllers were later derived via off-line multi-parametric MILP [9, 22]. Such controllers divide the space into polyhedral regions in which different linear controllers are optimal. The results for MLD systems extended earlier results for discrete-time linear time-invariant systems that were based on multi-parametric quadratic [14] and linear programming [10] for finding the optimal MPC controllers explicitly.

Unfortunately, although these methods provide an elegant solution to the general problem, they cannot handle the computational complexity of a typical trajectory planning problem and are not very useful in dynamic environments. Online optimization of a less general (i.e., smaller) problem for specific initial vehicle states is therefore of more interest. As demonstrated by the results in [116] and [129], thanks to the increase in computer speed and implementation of powerful state of the art algorithms in software packages such as CPLEX [56], MILP has become a feasible option for real-time path planning. Throughout this dissertation, we will provide more references to ongoing research and work on related topics and alternative approaches that was done in parallel at MIT and other universities.

1.3 Statement of Contributions

This thesis presents several new concepts for receding horizon trajectory planning of both single and multiple vehicles. The majority of the ideas and algorithms that are introduced can be considered independent of the underlying optimization method and as such can be generally used with path planning techniques other than MILP. Furthermore, some of the concepts fit within the broader field of MPC of hybrid systems. In this dissertation, however, we restrict ourselves to guidance problems and choose to illustrate the theory using MILP as the implementation framework. The contributions of this thesis can then be stated as follows:

- We introduce a general hybrid dynamic model for trajectory planning of highly agile vehicles. The proposed control architecture combines a velocity control system consisting of various continuous linear time-invariant modes with a discrete set of agile maneuvers. The latter are represented by affine transformations in the state space and can be described using a limited number of parameters such as the maneuver duration, displacement, and ingress and egress conditions such as entrance and exit velocity. The model efficiently captures the agile, nonlinear capabilities of the vehicle in a way that is well-suited for real-time trajectory planning using MILP. Compared to the maneuver automaton from [35], the required library of LTI modes and agile maneuvers is much smaller, more precise navigation is possible and obstacle avoidance constraints can be directly included in the trajectory optimization. We specialize the approach to a model of a small-scale rotorcraft based on MIT's aerobatic X-Cell helicopter.

- We extend the principle of receding horizon trajectory planning by including feasibil-

ity and safety guarantees in the problem formulation. We introduce the concept of a terminal feasible invariant set in which the vehicle can safely remain for an indefinite period of time. These sets are expressed as collections of affine transformations of the last state in the planning horizon and as such are computed online. Constraining the receding horizon trajectory computed at each iteration to terminate in such an invariant set guarantees nominal feasibility of the optimization problem at all future time steps. Safety is ensured by maintaining an *a priori* known backtrack trajectory that is updated at each receding horizon iteration. The feasibility and safety constraints are essential when maneuvering through environments that are only partially characterized and further explored online. Unlike other approaches such as [4, 146] our formulation does not require intensive off-line computations of fixed invariant sets. As such, it allows for more flexible and less conservative solutions for maintaining vehicle safety through cluttered environments.

- We use the terminal feasible invariant set concept for single vehicles to develop a new and fast algorithm for provably safe distributed trajectory planning for multiple vehicles. All vehicles that are within a certain conflict zone of each other will subsequently update and broadcast their paths. Each vehicle thereby only plans its own trajectory using a receding horizon strategy that accounts for the latest plans of all other vehicles in the conflict zone. The algorithm is applied to multiple aircraft where future feasibility of the planning problem is guaranteed by maintaining dynamically feasible trajectories for all aircraft that terminate in non-intersecting loiter patterns. Besides maintaining feasibility, if the problem is too complex to be solved within the time constraints of a real-time system, our approach also provides *a priori* safe rescue solutions for each vehicle. The algorithm is also applicable to maintaining nominal feasibility of a general distributed planning problem and can be used with any trajectory optimization technique that has collision avoidance guarantees. We present a MILP implementation and corresponding collision avoidance constraints.

- A proof-of-concept application is given of MILP-based multi-vehicle receding horizon path planning with feasibility guarantees. The problem of interest is to maintain wireless communication between a ground station and a vehicle that is performing a mission in a cluttered environment. Relay agents are therefore introduced that must be positioned throughout the environment in such a way that an indirect line of sight connection with the ground station is always maintained. We present a centralized and a distributed receding horizon algorithm that achieves such cooperation in a way that is more flexible than existing approaches. Feasibility at future time steps is ensured by using hover states as terminal feasible invariant sets. Obstacle avoidance, connectivity and collision avoidance constraints are again formulated using MILP. A first-time flight-test of multi-vehicle online MILP-based trajectory planning with two helicopters through a real obstacle environment was performed.

- Successful implementation and flight-test demonstration of single vehicle MILP-based receding horizon trajectory planning were done using an autonomous T-33 aircraft. In cooperation with Boeing Phantom Works and as part of DARPA's Software Enabled Control Program, a mission was flown through a partially unknown environment. Obstacle-free trajectories with feasibility guarantees were computed on-board the vehicle in real-time, adapting the flight path to pop-up no-fly zones and changing mission tasks.

- Numerous improvements to the basic MILP path planning formulation as introduced in [127, 125] are made. An efficient representation of arbitrarily shaped non-convex obstacles is developed that requires fewer binary variables and inequalities. Several vehicle models with new kino-dynamic constraints are introduced that capture helicopter and fixed-wing dynamics more precisely than the basic double integrator model. Alternative cost functions are presented for minimum time trajectories that yield comparable results but are much faster than exact shortest time formulations and allow for shorter planning horizons. Finally, several simplifications and heuristics are discussed that reduce the computational requirements in practical implementations.

1.4 Thesis Outline

The dissertation is organized as follows. Chapter 2 formalizes the basic receding horizon trajectory planning problem for single and multiple vehicles. The corresponding MILP formulation and various example scenarios for navigation through a cluttered environment are given. Chapter 3 then presents a hybrid control architecture for agile vehicles that enables inclusion of nonlinear maneuvers in the receding horizon optimization problem. Next, Chapter 4 extends the basic receding horizon strategy to include feasibility and safety constraints. It introduces the concept of a terminal feasible invariant set that is computed online and should be reachable from all states along the trajectory. Various simulation scenarios in which the feasibility and/or safety constraints prove to be essential are presented. Chapter 5 then uses the terminal feasible invariant set principle for a single vehicle to construct a safe distributed planning algorithm for multiple vehicles. A detailed description of the algorithm along with a formal feasibility proof and simulation results for multiple aircraft are given. In Chapter 6, the concepts of the previous chapters are combined and applied to the problem of multi-vehicle path planning for maintaining indirect line of sight communication between vehicles in a cluttered environment. Simulation, hardware in the loop and flight-test results of a two-helicopter mission are given. Next, Chapter 7 presents another flight-test demonstration involving a piloted F-15 and an autonomous T-33 aircraft guided by a MILP-based trajectory planner. An overview of the software architecture and natural language interface between the vehicles is given, and details and results of the MILP-based guidance system are discussed. Chapter 8 then concludes the thesis and outlines some topics for future work.

Chapter 2

Receding Horizon Trajectory Planning

This chapter presents the basic trajectory planning problem for single and multiple vehicles using a receding horizon planning strategy. A high-level mathematical problem statement is given in the form of an online optimization problem for which a mixed-integer programming implementation is worked out in detail. Constraints for general obstacle and collision avoidance are presented and a cost function is introduced that automatically switches from an approximate to an exact minimum time objective once the goal is within reach. The generality and flexibility of the MILP-based trajectory planning approach are illustrated through several single and multi-vehicle scenarios.

2.1 Introduction

As discussed in Chapter 1, over the last decade, both civilian and military institutions have expressed increased interest in the use of fully autonomous aircraft and helicopters or so-called unmanned aerial vehicles (UAVs) [101]. Such systems need no, or minor, human control from a ground station, thereby reducing operating costs and enabling missions in harsh or remote environments. A significant part of the vehicle autonomy consists of its path planning capabilities: the problem is to guide the vehicle through an obstacle field while accounting for its dynamic and kinematic properties.

In many applications, a detailed map of the environment is not available ahead of time, and obstacles are detected while the mission is carried out. In this chapter, we consider scenarios where the environment is only known within a certain detection radius around the vehicle. We assume that within that region, the environment is static and fully characterized. The knowledge of the environment could either be gathered through the detection capabilities of the vehicle itself, or result from cooperation with another, more sophisticated agent [94, 149].

Since the environment is explored online, a trajectory from a start to a destination location typically needs to be computed gradually over time, i.e., while the mission unfolds. This calls for a receding horizon strategy, in which a new segment of the total path is computed at each time step by solving a constrained optimization problem over a limited horizon. In [125, 127] a receding horizon approach based on mixed-integer linear programming was introduced that provides hard obstacle and collision avoidance guarantees, and allows inclusion of other non-convex state and input constraints. The basic MILP receding

horizon formulation presented in [127] was extended to account for local minima in [7]. In the latter, a cost-to-go function was introduced based on a graph representation of the whole environment between start and end point that guaranteed stability in the sense of reaching the goal. The approach has been further extended to account for turn rate constraints in [68] and for planning through three dimensional environments in [69].

In this chapter, however, we assume that the environment is *not* fully characterized before the mission. As such, a visibility graph as in [7] can only be constructed locally using a line of sight approximation of the distance to the goal beyond the detection radius. Such a heuristic will be used in Chapter 7. Here we will instead return to the basic formulation and focus on the dynamic, kinematic, obstacle avoidance and collision avoidance constraints. An extended MILP formulation for avoidance of arbitrarily shaped non-convex obstacles and a new switching cost function for guiding a vehicle to its goal state in a fast way are introduced.

2.2 Problem Formulation

2.2.1 Problem Setup

This section presents the basic trajectory planning problem for a team of vehicles that was outlined in Chapter 1 more formally. Let the various agents be denoted by an index $i = 1, \ldots, V$. For optimization purposes, the dynamics of each vehicle are characterized by discrete-time, linear state space models as follows:

$$\boldsymbol{x}_i(t+1) = \boldsymbol{A}_i \boldsymbol{x}_i(t) + \boldsymbol{B}_i \boldsymbol{u}_i(t), \ i = 1, \ldots, V \tag{2.1}$$

where $\boldsymbol{x}_i(t) \in \mathbb{R}^{N_x}$ is the state vector and $\boldsymbol{u}_i(t) \in \mathbb{R}^{N_u}$ is the input vector at the t^{th} time step. The state vector $\boldsymbol{x}_i(t)$ is typically made up of the position and velocity in a 3D inertial coordinate frame (east, north, altitude), respectively denoted by $\boldsymbol{p}_i(t) \equiv [x_i(t)\ y_i(t)\ z_i(t)]' \in \mathbb{R}^3$ and $\boldsymbol{v}_i(t) \equiv [\dot{x}_i(t)\ \dot{y}_i(t)\ \dot{z}_i(t)]' \in \mathbb{R}^3$. A trajectory will then consist of a sequence of states $\boldsymbol{x}_i(t) \equiv [\boldsymbol{p}_i'(t)\ \boldsymbol{v}_i'(t)]'$ or generalized waypoints that the vehicle must follow.

Depending on the particular model, the input vector $\boldsymbol{u}_i(t)$ is a 3D inertial acceleration or reference velocity vector. In both cases, however, combined with additional constraints on $\boldsymbol{x}_i(t)$ and $\boldsymbol{u}_i(t)$, the state space model (2.1) must capture the closed-loop dynamics that result from augmenting the vehicle with a waypoint or velocity tracking controller. These constraints should capture kinematic and dynamic properties such as maximum speed, acceleration and turn rate, and will be denoted as follows: $\boldsymbol{x}_i(t) \in \mathcal{X}_i(t)$ and $\boldsymbol{u}_i(t) \in \mathcal{U}_i(t)$. Note that the constraint sets $\mathcal{X}_i(t)$ and $\mathcal{U}_i(t)$ are time-dependent, which accommodates the use of robust planning approaches such as constraint tightening [46, 115].

Accounting for the vehicle dynamics in the trajectory planning problem will ensure that the trajectories are dynamically feasible, i.e., that the vehicles can execute or track them. Besides the dynamics, feasibility will be affected by the presence of obstacles in the environment, such as buildings, hills or other no-fly zones. We define the set $\mathcal{O}_i \subset \mathbb{R}^3$ of all obstacles that are relevant to vehicle i as the regions in the inertial space that the vehicle is forbidden to enter: $\boldsymbol{p}_i(t) \notin \mathcal{O}_i$. By "relevant" we mean that the sets $\mathcal{O}_i$ include all obstacles that are located within the distance that is reachable from the initial position of vehicle i over one planning horizon. If the environment within that radius is only partially-known, the unknown areas should be modeled as obstacles too. The features of the environment beyond this planning radius are irrelevant for the trajectory optimization at time t.

As is common practice in the field of robot motion planning, the actual obstacles are enlarged with the dimensions of the vehicle, such that the vehicle itself can be treated as a point in this so-called configuration space [72, 52]. We summarize as follows:

Definition 2.1 (Obstacle Avoidance): *We say that there is* obstacle avoidance *for vehicle i if $\boldsymbol{p}_i(t) \notin \mathcal{O}_i$, where $\boldsymbol{p}_i(t) \in \mathbb{R}^3$ is the inertial position and $\mathcal{O}_i \subset \mathbb{R}^3$ represents the set of all forbidden regions in the environment that are relevant for vehicle i. The members of $\mathcal{O}_i$ are the actual obstacles enlarged with the largest dimension of the vehicle.*

Furthermore, in order to avoid collisions, the vehicles should remain at a safe distance d_{safe} from each other at all time. In general, this distance may be different for each vehicle, but for simplicity of notation, we consider it the same for all. The collision avoidance requirement can thus be expressed as follows:

Definition 2.2 (Collision Avoidance): *We say that there is* collision avoidance *if the Euclidean separation distance between all pairs of vehicles $(i, j),\ i = 1, \ldots, V-1,\ j > i$, is greater than or equal to a certain safety margin d_{safe}: $d_{ij} = \|\boldsymbol{p}_i - \boldsymbol{p}_j\| \geq d_{safe}$.*

The overall goal of the trajectory planning problem is then for the vehicles to perform a certain task while satisfying the preceding constraints and optimizing an associated cost. The cost function can be a measure of time, fuel or a more sophisticated criterion such as visibility w.r.t. a threat or radar. The task for each vehicle i will typically be specified as having to fly to a certain waypoint of interest, denoted as the final state $\boldsymbol{x}_{f,i} \equiv [\boldsymbol{p}'_{f,i}\ \boldsymbol{v}'_{f,i}]'$. This waypoint could be an intermediate state along a more elaborate flight plan or mission that was designed by a higher level decision unit [5, 89, 130]. In that case, the path planner will consider the subsequent waypoints as a series of independent tasks that are only coupled by the entry/exit conditions at each waypoint. As such, we are assuming that the task assignment and trajectory planning problems are decoupled, which – for reasons of computational tractability– is standard practice in the UAV and robotics literature [6, 78, 113]. An example of how a choice of target states can be incorporated in the optimization problem can be found in [117]. Finally, in this chapter, we focus on a centralized approach, in which one entity (e.g., a ground station) computes trajectories for all vehicles simultaneously. An extension to distributed strategies where each agent optimizes its own trajectory is presented in Chapter 5.

2.2.2 Receding Horizon Planning

Depending on the number of vehicles and the distance they have to travel, computing complete trajectories from start to finish at once might be computationally too expensive. Indeed, it is known that, even for a single vehicle, motion planning is a PSPACE-hard problem [110, 49]. Moreover, the environment is only partially-known and further explored in real-time. The trajectories will therefore have to be computed gradually over time while the mission unfolds. This can be accomplished using an online receding horizon strategy, in which partial trajectories from the current states $\boldsymbol{x}_i(t)$ towards the goal states $\boldsymbol{x}_{f,i}$ are computed by solving the trajectory optimization problem over a limited horizon of T time steps. This provides a sequence of T new states/waypoints and corresponding control inputs for each of the vehicles. However, only a subset of the control sequence is actually implemented: e.g., only the first waypoint of each vehicle is given to the respective waypoint controllers, or the first velocity command is executed. The process is then repeated at the next time step $t + 1$, and so on until the vehicles reach their respective goals. As such, new

measurements of the states of the vehicles and new information about the environment can be taken into account at each iteration.

Let the sequence of T steps starting at time t be denoted by indices $(t+k|t)$. For each vehicle $i = 1, \ldots, V$, the corresponding state and control sequence is then given by $\boldsymbol{x}_i(t+k|t),\ k = 0, \ldots, T$ and $\boldsymbol{u}_i(t+k|t),\ k = 0, \ldots, T-1$. Because of the computation delay, however, the trajectories starting at time step t must be computed during time step $t-1$, i.e., when the helicopters are on their way to the initial states $\boldsymbol{x}_i(t+0|t)$ of the new optimization problem. Hence, the initial states $\boldsymbol{x}_i(t|t)$ should be predictions $\hat{\boldsymbol{x}}_i(t|t-1) = [\hat{\boldsymbol{p}}_i'(t|t-1)\ \hat{\boldsymbol{v}}_i'(t|t-1)]'$ made during the previous time step $t-1$ of what the position and velocity of the vehicles will be when the plan is actually implemented at the start of time step t. In the nominal case, where no disturbances are acting on the vehicles and there are no uncertainties in the dynamic model (2.1), the predicted state is identical to the first state $\boldsymbol{x}_i(t|t-1)$ of the previous plan: $\boldsymbol{x}_i(t|t) = \hat{\boldsymbol{x}}_i(t|t-1) \equiv \boldsymbol{x}_i(t|t-1)$. We will make this assumption throughout the remainder of this and the next chapters. Alternatively, this is equivalent to assuming that the vehicles are equipped with accurate waypoint controllers that can exactly track the desired trajectories. Including robustness against disturbances could be done using constraint tightening methods [46, 71, 112]; accounting for uncertainties in the vehicle models is still a topic of ongoing research.

2.2.3 Optimization Problem

Multiple Vehicles

As discussed previously, the primary objective of the trajectory optimization problem is to guide the vehicles to their goal states $\boldsymbol{x}_{f,i}$, thereby optimizing a certain performance criterion and avoiding obstacles and collisions. To capture the actions of all vehicles simultaneously, we introduce an objective function J_T of the following form:

$$J_T = \sum_{i=1}^{V} \sum_{k=0}^{T-1} \ell_{i,k}(\boldsymbol{x}_i(t+k|t), \boldsymbol{u}_i(t+k|t), \boldsymbol{x}_{f,i}) + f_{T,i}(\boldsymbol{x}_i(t+T|t), \boldsymbol{x}_{f,i}) \tag{2.2}$$

in which $\ell_{i,k}(\cdot)$ indicates the stage cost associated with vehicle i at the k^{th} time step, and $f_{T,i}(\cdot)$ represents a terminal cost function.

According to Bellman's principle of optimality [8], the ideal terminal cost is the exact cost-to-go (e.g., time-to-go) from the last state $\boldsymbol{x}_i(t+T|t)$ in the planning horizon to the desired state $\boldsymbol{x}_{f,i}$. However, computing the exact cost-to-go generally requires solving a fixed horizon problem from the last state in the planning horizon, thus defeating the benefits of using a receding horizon policy. Moreover, the exact cost-to-go might be unknown if the environment is not fully characterized. Hence, heuristic methods are often used to generate estimates of the cost-to-go that guarantee some form of stability (e.g., reaching the goal without getting trapped in local minima of the cost function) and some level of performance (e.g., bounds on suboptimality) [7, 91].

The centralized multi-vehicle trajectory optimization problem at time t can now be formulated as:

$$J_T^* = \min_{\boldsymbol{x}_i(\cdot), \boldsymbol{u}_i(\cdot)} \sum_{i=1}^{V} \sum_{k=0}^{T-1} \ell_{i,k}(\boldsymbol{x}_i(t+k|t), \boldsymbol{u}_i(t+k|t), \boldsymbol{x}_{f,i}) + f_{T,i}(\boldsymbol{x}_i(t+T|t), \boldsymbol{x}_{f,i}) \tag{2.3}$$

subject to: $\forall i = 1, \ldots, V$:

$$\begin{aligned}
\boldsymbol{x}_i(t+k+1|t) &= \boldsymbol{A}_i\boldsymbol{x}_i(t+k|t) + \boldsymbol{B}_i\boldsymbol{u}_i(t+k|t),\ k = 0, \ldots, T-1 && (2.4)\\
\boldsymbol{x}_i(t|t) &= \hat{\boldsymbol{x}}_i(t|t-1) && (2.5)\\
\boldsymbol{x}_i(t+k|t) &\in \mathcal{X}_i(k),\ k = 1, \ldots, T && (2.6)\\
\boldsymbol{u}_i(t+k|t) &\in \mathcal{U}_i(k),\ k = 0, \ldots, T-1 && (2.7)\\
\boldsymbol{p}_i(t+k|t) &\notin \mathcal{O}_{a,i}(t),\ k = 1, \ldots, T && (2.8)\\
\|\boldsymbol{p}_i(t+k|t) - \boldsymbol{p}_j(t+k|t)\| &\geq d_{safe},\ j \geq i+1,\ k = 1, \ldots, T && (2.9)
\end{aligned}$$

Since the problem only makes sense if the initial states $\boldsymbol{x}_i(t|t)$ are feasible, the state constraints on the first time step were removed: if they did not hold, the optimization problem would be infeasible from the start. Furthermore, to prevent the discrete-time trajectories from cutting corners of obstacles in between two time steps, the obstacle sets $\mathcal{O}_{a,i}$ contain the actual obstacles enlarged with a safety envelope. The trajectories may then cut through the envelope instead, but will avoid the actual obstacles.

Single Vehicle

For a single vehicle, the trajectory optimization (2.3)-(2.9) reduces to the following problem:

$$J_T^* = \min \sum_{k=0}^{T-1} \ell_k(\boldsymbol{x}(t+k|t), \boldsymbol{u}(t+k|t), \boldsymbol{x}_f) + f_T(\boldsymbol{x}(t+T|t), \boldsymbol{x}_f) \qquad (2.10)$$

subject to:

$$\begin{aligned}
\boldsymbol{x}(t+k+1|t) &= \boldsymbol{A}\boldsymbol{x}(t+k|t) + \boldsymbol{B}\boldsymbol{u}(t+k|t),\ k = 0, \ldots, T-1 && (2.11)\\
\boldsymbol{x}(t|t) &= \hat{\boldsymbol{x}}(t|t-1) && (2.12)\\
\boldsymbol{x}(t+k|t) &\in \mathcal{X}(k),\ k = 1, \ldots, T && (2.13)\\
\boldsymbol{u}(t+k|t) &\in \mathcal{U}(k),\ k = 0, \ldots, T-1 && (2.14)\\
\boldsymbol{p}(t+k|t) &\notin \mathcal{O}_a(t),\ k = 1, \ldots, T && (2.15)
\end{aligned}$$

which now only contains obstacle avoidance constraints.

2.3 MILP Formulation

The optimization problem outlined above lends itself well to be formulated as a mixed-integer linear program. Mixed-integer linear programming (MILP) is a powerful optimization framework that allows inclusion of integer variables and discrete logic in a continuous linear optimization problem. An overview of applications and algorithms to solve such problems can be found in [33]. MILP is still an active research topic in operations research [144, 152], and the state of the art in cutting plane methods, integral basis methods, enumerative methods, and approximation algorithms is covered in the recent book by Bertsimas and Weismantel [19].

2.3.1 Mixed Integer Linear Programming

As an illustration of how logical decisions can be incorporated in an optimization problem, consider the following example. Assume that a cost function $J(\boldsymbol{x})$ needs to be minimized subject to either one of two constraints $\ell_1(\boldsymbol{x})$ and $\ell_2(\boldsymbol{x})$ on the continuous decision vector $\boldsymbol{x}$:

$$\begin{aligned} &\min_{\boldsymbol{x}} J(\boldsymbol{x}) \\ &\text{subject to:} \\ &\qquad \ell_1(\boldsymbol{x}) \;\leq\; 0 \\ &\text{OR} \quad \ell_2(\boldsymbol{x}) \;\leq\; 0 \end{aligned} \tag{2.16}$$

By introducing a large positive number M and a binary variable b, this optimization problem can equivalently be formulated as follows:

$$\begin{aligned} &\min_{\boldsymbol{x}} J(\boldsymbol{x}) \\ &\text{subject to:} \\ &\qquad \ell_1(\boldsymbol{x}) \;\leq\; Mb \\ &\text{AND} \quad \ell_2(\boldsymbol{x}) \;\leq\; M(1-b) \\ &\qquad\qquad b \;\in\; \{0,1\} \end{aligned} \tag{2.17}$$

When $b = 0$, constraint $\ell_1(\boldsymbol{x})$ must be satisfied, whereas $\ell_2(\boldsymbol{x})$ is relaxed. Namely, if M is chosen sufficiently large, $\ell_2(\boldsymbol{x}) \leq M(1-b)$ is always satisfied independent of the value of $\boldsymbol{x}$. The situation is reversed when $b = 1$. Since b can only take the binary values 0 or 1, at least one of the constraints $\ell_1(\boldsymbol{x})$ and $\ell_2(\boldsymbol{x})$ will be satisfied, which is equivalent to the original "OR"-formulation (2.16). In the special case where $J(\boldsymbol{x})$, $\ell_1(\boldsymbol{x})$ and $\ell_2(\boldsymbol{x})$ are (affine) linear expressions, problem (2.17) is a MILP.

The formulation can easily be extended to account for multiple constraints $\ell_k(\boldsymbol{x}),\ k = 1,\ldots,K$, out of which at least N must be satisfied simultaneously. This is done as follows:

$$\begin{aligned} &\min_{\boldsymbol{x}} J(\boldsymbol{x}) \\ &\text{subject to:} \\ &\quad \ell_k(\boldsymbol{x}) \;\leq\; Mb_k,\ k = 1,\ldots,K \\ &\quad \sum_k b_k \;\leq\; K - N \\ &\qquad b_k \;\in\; \{0,1\} \end{aligned} \tag{2.18}$$

The additional summation constraint ensures that at least N of the binary variables b_k are 0, thus guaranteeing that at least N of the inequalities $\ell_k(\boldsymbol{x}) \leq 0$ are satisfied simultaneously. More generally, using a vector $\boldsymbol{b}$ of binary variables, any polyhedron or intersection of polyhedra described by linear constraints on a continuous decision vector $\boldsymbol{x}$ can then be described as follows:

$$\boldsymbol{L}\boldsymbol{x} + \boldsymbol{M}\boldsymbol{b} + \boldsymbol{k} \leq 0 \tag{2.19}$$

The constant matrix $\boldsymbol{M}$ and vector $\boldsymbol{k}$ contain large numbers M and integer constants K and N that can encode any binary logic such as that of problem (2.18).

2.3.2 Vehicle Dynamics

In the trajectory planning problem, the continuous optimization is done over the states and inputs, while binary variables are introduced to capture non-convex constraints such

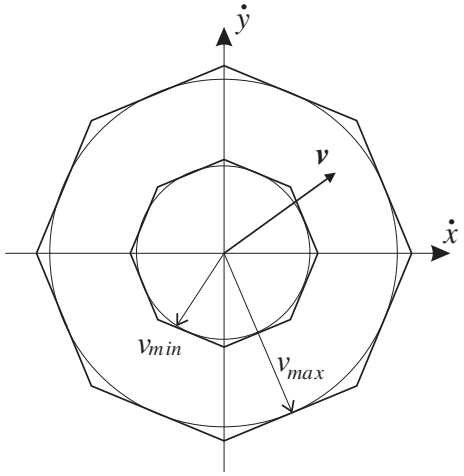


Figure 2-1: Approximation of minimum and maximum velocity bounds by polygons.

as obstacle avoidance, collision avoidance and minimum velocity requirements. Since the structure of the optimization problem is the same at every receding horizon iteration, in what follows, we will shorten the time step index $(t+k|k)$ to k to simplify the notation.

Although it is possible to use more complicated models, for the basic problem presented in this chapter, it is sufficient to approximate the vehicle dynamics by a double integrator model with constraints on speed and acceleration. Alternative models will be introduced in later chapters. The 3D discrete-time, unconstrained double integrator dynamics used for now are the following:

$$\begin{bmatrix} \boldsymbol{p}_i(k+1) \\ \boldsymbol{v}_i(k+1) \end{bmatrix} = \begin{bmatrix} \boldsymbol{I}_3 & \Delta t \boldsymbol{I}_3 \\ \boldsymbol{O}_3 & \boldsymbol{I}_3 \end{bmatrix} \begin{bmatrix} \boldsymbol{p}_i(k) \\ \boldsymbol{v}_i(k) \end{bmatrix} + \begin{bmatrix} \frac{(\Delta t)^2}{2} \boldsymbol{I}_3 \\ \Delta t \boldsymbol{I}_3 \end{bmatrix} \boldsymbol{a}_i(k) \tag{2.20}$$

where $\boldsymbol{a}_i(k) \equiv [\ddot{x}_i(k)\ \ddot{y}_i(k)\ \ddot{z}_i(k)]'$ is the inertial acceleration vector, Δt is the time discretization step, and $\boldsymbol{I}_3$ and $\boldsymbol{O}_3$ represent identity and zero matrices of size 3x3.

We use the above dynamic model for both fixed-wing aircraft and rotorcraft scenarios. In the fixed-wing case, however, we will only consider 2D scenarios with corresponding 2D double integrator dynamics. Limiting the magnitude of the planar velocity and acceleration vectors can then be accomplished by approximating their 2-norms $\|[\dot{x}_i(k)\ \dot{y}_i(k)]'\|$ and $\|[\ddot{x}_i(k)\ \ddot{y}_i(k)]'\|$ by the edges of an N-sided polygon (see Figure 2-1). This yields the following set of linear inequalities:

$$\dot{x}_i(k)\sin\left(\frac{2\pi n}{N}\right) + \dot{y}_i(k)\cos\left(\frac{2\pi n}{N}\right) \leq v_{max,i},\ n = 1,\ldots,N,\ k = 1,\ldots,T \tag{2.21}$$

$$\ddot{x}_i(k)\sin\left(\frac{2\pi n}{N}\right) + \ddot{y}_i(k)\cos\left(\frac{2\pi n}{N}\right) \leq a_{max,i},\ n = 1,\ldots,N,\ k = 0,\ldots,T-1 \tag{2.22}$$

For coordinated turns, inequalities (2.22) also implicitly express a constraint on the maximum turn rate $\omega_{max,i}$: it is limited to $a_{max,i}/v_{max,i}$ at the maximum speed and to $a_{max,i}/v_{min,i}$ at the minimum speed [114].

Since a fixed-wing aircraft has to produce enough lift to keep flying, it must maintain a minimum velocity v_{min}. As shown in Figure 2-1, this requirement can be expressed by

forcing the velocity vector to lie *outside* a circle of radius v_{min}. By again approximating the circle by an N-sided polygon, this non-convex constraint can be captured by the following inequalities [125, 114]:

$$\dot{x}_i(k)\sin\left(\frac{2\pi n}{N}\right)+\dot{y}_i(k)\cos\left(\frac{2\pi n}{N}\right) \geq v_{min,i}-Mc_{in}(k),\ n=1\ldots N,\ k=1\ldots T \quad (2.23)$$

$$\sum_{n=1}^{N} c_{in}(k) \leq N-1 \quad (2.24)$$

$$c_{in}(k) \in \{0,1\} \quad (2.25)$$

where $c_{in}(k)$ are binary variables and M is a sufficiently large constant number. As in problem (2.18), these constraints ensure that the velocity vector lies in at least one of the outer halfplanes defined by the edges of the polygon, as opposed to lying in all of the inner halfplanes as for the convex maximum speed constraint. Inequalities (2.21) and (2.23)-(2.24) on the one hand, and (2.22) on the other hand, then respectively express the non-convex state and convex input constraint sets $\mathcal{X}_i(k)$ and $\mathcal{U}_i(k)$ for the 2D fixed-wing UAV.

For the 3D helicopter case, the altitude dynamics can typically be considered decoupled from the planar x-y ones [39]. The limits on climb/descend rate and acceleration can then be expressed as follows:

$$\dot{z}_{min,i} \leq \dot{z}_i(k) \leq \dot{z}_{max,i},\quad k=1,\ldots,T \quad (2.26)$$

$$\ddot{z}_{min,i} \leq \ddot{z}_i(k) \leq \ddot{z}_{max,i},\quad k=0,\ldots,T-1 \quad (2.27)$$

Moreover, since a helicopter has the ability to hover, no minimum speed constraints must be accounted for in the trajectory planning problem. The state constraint sets $\mathcal{X}_i(k)$ are now given by inequalities (2.21) and (2.26), the input sets $\mathcal{U}_i(k)$ are formed by constraints (2.22) and (2.27).

2.3.3 Obstacle Avoidance

Let an index o denote the individual obstacles in $\mathcal{O}_{a,i} \subset \mathbb{R}^3$, with $\mathcal{O}_{a,i}$ the set of enlarged obstacles that are within reach of vehicle i at the current planning iteration. Since our objective is to use linear optimization techniques, an arbitrarily shaped obstacle o is first approximated by a (possibly non-convex) polyhedron $\mathcal{P}_{io}$. Next, we construct the convex hull of $\mathcal{P}_{io}$ and denote each of the resulting faces by an index e, $e=1,\ldots,H_{io}$. A sufficient condition for obstacle avoidance is then that all points along the planned trajectory of vehicle i lie outside this convex hull, i.e., in at least one of the outer halfspaces determined by the faces e. Let these halfspaces be described by $u_{ioe}x+v_{ioe}y+w_{ioe}z+h_{ioe}\leq 0,\ e=1,\ldots,H_{io}$. Avoidance of obstacle o by vehicle i can then be expressed as:

$$\begin{array}{rl} \forall k=1,\ldots,T: & u_{io1}x_i(k)+v_{io1}y_i(k)+w_{io1}z_i(k)+h_{io1} \leq 0 \\ \text{OR} & u_{io2}x_i(k)+v_{io2}y_i(k)+w_{io2}z_i(k)+h_{io2} \leq 0 \\ \vdots & \vdots \\ \text{OR} & u_{ioH_{io}}x_i(k)+v_{ioH_{io}}y_i(k)+w_{ioH_{io}}z_i(k)+h_{ioH_{io}} \leq 0 \end{array} \quad (2.28)$$

To capture these logical OR constraints in a mathematical fashion, we introduce binary variables $b_{ioe}(k) \in \{0, 1\},\ e = 1, \ldots, H_{io}$, and modify expressions (2.28) as follows:

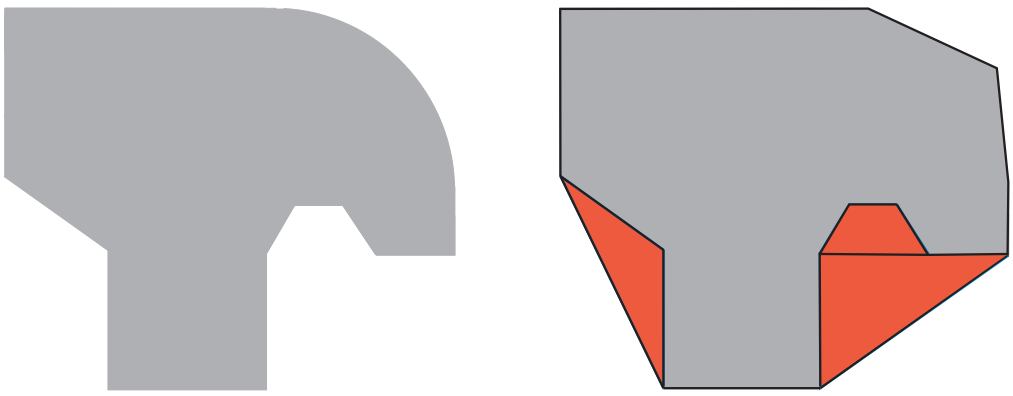
$$\begin{array}{rrcl} \forall k = 1, \ldots, T: & u_{io1}x_i(k) + v_{io1}y_i(k) + w_{io1}z_i(k) + h_{io1} & \leq & Mb_{io1}(k) \\ \text{AND} & u_{io2}x_i(k) + v_{io2}y_i(k) + w_{io2}z_i(k) + h_{io2} & \leq & Mb_{io2}(k) \\ \vdots & & \vdots & \\ \text{AND} & u_{ioH_{io}}x_i(k) + v_{ioH_{io}}y_i(k) + w_{ioH_{io}}z_i(k) + h_{ioH_{io}} & \leq & Mb_{ioH_{io}}(k) \\ \text{AND} & \sum_{e=1}^{H_{io}} b_{ioe}(k) & \leq & H_{io} - 1 \end{array} \tag{2.29}$$

Here M is again an arbitrary constant that is larger than any of the values the left-hand sides of the inequalities can take in the current planning problem. Then, if $b_{ioe}(k) = 1$ for a particular e, the corresponding inequality is relaxed and always satisfied. As in problem (2.18), the last constraint ensures that at least one of the binaries $b_{ioe}(k)$ is 0, such that at least one of the original OR inequalities holds and obstacle avoidance is guaranteed.

For example, for a rectangular obstacle that is aligned with the xyz-coordinate frame, the avoidance constraints can be expressed as:

$$\begin{array}{rrcrcl} \forall k = 1, \ldots, T: & x_i(k) & \leq & x_{o,min} & + & Mb_{io1}(k) \\ & y_i(k) & \leq & y_{o,min} & + & Mb_{io2}(k) \\ & z_i(k) & \leq & z_{o,min} & + & Mb_{io3}(k) \\ & -x_i(k) & \leq & -x_{o,max} & + & Mb_{io4}(k) \\ & -y_i(k) & \leq & -y_{o,max} & + & Mb_{io5}(k) \\ & -z_i(k) & \leq & -z_{o,max} & + & Mb_{io6}(k) \\ & \sum_{e=1}^{6} b_{ioe}(k) & \leq & 5 & & \end{array} \tag{2.30}$$

where $(x_{o,min}, y_{o,min}, z_{o,min})$ and $(x_{o,max}, y_{o,max}, z_{o,max})$ respectively denote the vertices



(a) Actual obstacle shape (b) Polytopic approximation

Figure 2-2: Approximation of an arbitrary 2D obstacle by its convex hull and with convex polygons filling up the concavities. The vehicle is allowed to be inside the red areas, but must stay outside the grey area.

with the smallest and largest coordinates in each direction. In case $z_{o,min}$ equals the ground level z_0, the corresponding inequality can be dropped and replaced by $z_i(k) \geq z_0$. Accordingly, the sum of the 5 remaining binaries should then be less than 4.

If the surrounding polyhedron $\mathcal{P}_{io}$ is nonconvex, trajectory points can also lie inside concavities that are reachable from the outside. By partitioning each concavity into convex polyhedral parts c, $c = 1, \ldots, C_{io}$, an additional binary $\hat{b}_{ioc}(k)$ for each of these parts can be introduced that captures several inequalities at once. Figure 2-2 shows an example of such a partitioning. Let the faces g and corresponding inner halfspaces of each of the convex parts c be described by $\hat{u}^g_{ioc}x + \hat{v}^g_{ioc}y + \hat{w}^g_{ioc}z + \hat{h}^g_{ioc} \leq 0,\ g = 1, \ldots, G_{ioc}$. The avoidance logic (2.28) can then be extended as follows:

$$
\begin{array}{rlrcl}
\forall k = 1, \ldots, T: & & u_{io1}x_i(k) + v_{io1}y_i(k) + w_{io1}z_i(k) + h_{io1} & \leq & 0 \\
\text{OR} & & u_{io2}x_i(k) + v_{io2}y_i(k) + w_{io2}z_i(k) + h_{io2} & \leq & 0 \\
\vdots & & & \vdots & \\
\text{OR} & & u_{ioH_{io}}x_i(k) + v_{ioH_{io}}y_i(k) + w_{ioH_{io}}z_i(k) + h_{ioH_{io}}(k) & \leq & 0 \\
\text{OR} & & \hat{u}^1_{io1}x_i(k) + \hat{v}^1_{io1}y_i(k) + \hat{w}^1_{io1}z_i(k) + \hat{h}^1_{io1} & \leq & 0 \\
 & \text{AND} & \hat{u}^2_{io1}x_i(k) + \hat{v}^2_{io1}y_i(k) + \hat{w}^2_{io1}z_i(k) + \hat{h}^2_{io1} & \leq & 0 \\
 & \vdots & & \vdots & \\
 & \text{AND} & \hat{u}^{G_{io1}}_{io1}x_i(k) + \hat{v}^{G_{io1}}_{io1}y_i(k) + \hat{w}^{G_{io1}}_{io1}z_i(k) + \hat{h}^{G_{io1}}_{io1} & \leq & 0 \\
\vdots & & & \vdots & \\
\text{OR} & & \hat{u}^1_{ioC_{io}}x_i(k) + \hat{v}^1_{ioC_{io}}y_i(k) + \hat{w}^1_{ioC_{io}}z_i(k) + \hat{h}^1_{ioC_{io}} & \leq & 0 \\
 & \text{AND} & \hat{u}^2_{ioC_{io}}x_i(k) + \hat{v}^2_{ioC_{io}}y_i(k) + \hat{w}^2_{ioC_{io}}z_i(k) + \hat{h}^2_{ioC_{io}} & \leq & 0 \\
 & \vdots & & \vdots & \\
 & \text{AND} & \hat{u}^{G_{ioC_{io}}}_{ioC_{io}}x_i(k) + \hat{v}^{G_{ioC_{io}}}_{ioC_{io}}y_i(k) + \hat{w}^{G_{ioC_{io}}}_{ioC_{io}}z_i(k) + \hat{h}^{G_{ioC_{io}}}_{ioC_{io}} & \leq & 0
\end{array}
\tag{2.31}
$$

The corresponding relaxed constraints become:

$$
\begin{array}{rlrcl}
\forall k = 1, \ldots, T: & & u_{io1}x_i(k) + v_{io1}y_i(k) + w_{io1}z_i(k) + h_{io1} & \leq & Mb_{io1}(k) \\
\text{AND} & & u_{io2}x_i(k) + v_{io2}y_i(k) + w_{io2}z_i(k) + h_{io2} & \leq & Mb_{io2}(k) \\
\vdots & & & \vdots & \\
\text{AND} & & u_{ioH_{io}}x_i(k) + v_{ioH_{io}}y_i(k) + w_{ioH_{io}}z_i(k) + h_{ioH_{io}}(k) & \leq & Mb_{ioH_{io}}(k) \\
\text{AND} & & \hat{u}^1_{io1}x_i(k) + \hat{v}^1_{io1}y_i(k) + \hat{w}^1_{io1}z_i(k) + \hat{h}^1_{io1} & \leq & M\hat{b}_{io1}(k) \\
 & \text{AND} & \hat{u}^2_{io1}x_i(k) + \hat{v}^2_{io1}y_i(k) + \hat{w}^2_{io1}z_i(k) + \hat{h}^2_{io1} & \leq & M\hat{b}_{io1}(k) \\
 & \vdots & & \vdots & \\
 & \text{AND} & \hat{u}^{G_{io1}}_{io1}x_i(k) + \hat{v}^{G_{io1}}_{io1}y_i(k) + \hat{w}^{G_{io1}}_{io1}z_i(k) + \hat{h}^{G_{io1}}_{io1} & \leq & M\hat{b}_{io1}(k) \\
\vdots & & & \vdots & \\
\text{AND} & & \hat{u}^1_{ioC_{io}}x_i(k) + \hat{v}^1_{ioC_{io}}y_i(k) + \hat{w}^1_{ioC_{io}}z_i(k) + \hat{h}^1_{ioC_{io}} & \leq & M\hat{b}_{ioC_{io}}(k) \\
 & \text{AND} & \hat{u}^2_{ioC_{io}}x_i(k) + \hat{v}^2_{ioC_{io}}y_i(k) + \hat{w}^2_{ioC_{io}}z_i(k) + \hat{h}^2_{ioC_{io}} & \leq & M\hat{b}_{ioC_{io}}(k) \\
 & \vdots & & \vdots & \\
 & \text{AND} & \hat{u}^{G_{ioC_{io}}}_{ioC_{io}}x_i(k) + \hat{v}^{G_{ioC_{io}}}_{ioC_{io}}y_i(k) + \hat{w}^{G_{ioC_{io}}}_{ioC_{io}}z_i(k) + \hat{h}^{G_{ioC_{io}}}_{ioC_{io}} & \leq & M\hat{b}_{ioC_{io}}(k) \\
\text{AND} & & \sum_{e=1}^{H_{io}} b_{ioe}(k) + \sum_{c=1}^{C_{io}} \hat{b}_{ioc}(k) & \leq & H_{io} + C_{io} - 1
\end{array}
\tag{2.32}
$$

The last inequality now ensures that at time step k, vehicle i will either be outside the convex hull or inside one of the concavities of obstacle o.

As discussed earlier, the actual obstacles are enlarged by the dimensions of the vehicle and an additional safety envelope that compensates for the discrete-time nature of the trajectories. The latter should have at least an xy-thickness of $(\sqrt{(}2)/2)v_{max}\Delta t$ and a z-dimension of $\max(\dot{z}_{max}, |\dot{z}_{min}|)\Delta t$. If this envelope is too large compared to the size of the obstacles, additional avoidance checks can be carried out for linearly interpolated positions between the waypoints along the trajectory. The vehicle coordinates $\boldsymbol{p}_i(k) \equiv [x_i(k)\, y_i(k)\, z_i(k)]'$ in constraints (2.28)-(2.32) should then be replaced by:

$$\boldsymbol{p}_i(k-1) + \frac{l}{L_a}\left(\boldsymbol{p}_i(k) - \boldsymbol{p}_i(k-1)\right),\ l = 1, \ldots, L_a \tag{2.33}$$

where L_a indicates the number of interpolation points and additional binary variables should be introduced accordingly.

2.3.4 Collision Avoidance

The collision avoidance constraints (2.9) are also non-convex and again require the use of binary variables. To reduce the complexity of the optimization problem, the 2-norm distance is approximated by the 1-norm. This comes down to considering an avoidance box of size $2d_{safe}$ around each vehicle, which must at least account for the distance the vehicles can cover within one time-step: $d_{safe} > v_{max}\Delta t$. The avoidance constraints for all pairs of vehicles $(i, j),\ j = 0, j \geq i+1$, can then be formulated as follows:

$$\begin{array}{rrcl} \forall k = 1, \ldots, T: & |x_i(k) - x_j(k)| & \geq & d_{safe} \\ & \text{OR} \quad |y_i(k) - y_j(k)| & \geq & d_{safe} \\ & \text{OR} \quad |z_i(k) - z_j(k)| & \geq & d_{safe} \end{array} \tag{2.34}$$

By introducing binaries $\hat{d}_{ijr}(k)$ for each pair (i, j) and each time step k, the above OR-constraints are transformed into:

$$\begin{array}{rrcl} \forall k = 1, \ldots, T: & -x_i(k) + x_j(k) & \leq & -d_{safe} + M\hat{d}_{ij1}(k) \\ & -x_j(k) + x_i(k) & \leq & -d_{safe} + M\hat{d}_{ij2}(k) \\ & -y_i(k) + y_j(k) & \leq & -d_{safe} + M\hat{d}_{ij3}(k) \\ & -y_j(k) + y_i(k) & \leq & -d_{safe} + M\hat{d}_{ij4}(k) \\ & -z_i(k) + z_j(k) & \leq & -d_{safe} + M\hat{d}_{ij5}(k) \\ & -z_j(k) + z_i(k) & \leq & -d_{safe} + M\hat{d}_{ij6}(k) \\ & \sum_{r=1}^{6} \hat{d}_{ijr}(k) & \leq & 5 \end{array} \tag{2.35}$$

where M is again a sufficiently large number [127].

2.4 Example Scenarios

2.4.1 Example 1: UAV in 2D

As an illustration of the MILP approach, consider the following 2D example of a small autonomous aircraft with the following parameters: $v_{max} = 4$ m/s, $v_{min} = 2$ m/s and

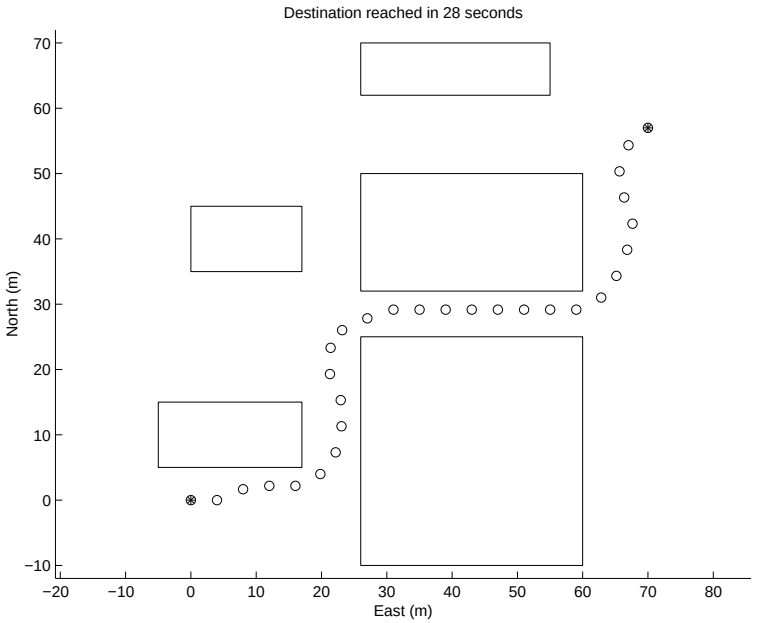


Figure 2-3: Example 1: The aircraft is initially in the origin, flying east at 4 m/s, and has to maneuver to position $(70, 57)$ m. The goal is reached after 28 s.

$\omega_{max} = 30$ deg/s corresponding to $a_{max} = \omega_{max} v_{max} = 2.09$ m/s^2. The planning horizon T contains 6 time steps of 1 s each and $N = 8$ for the polygonal approximation of the speed and acceleration constraints. The scenario is illustrated in Figure 2-3: the aircraft is initially in the origin, flying east at 4 m/s, and needs to maneuver to position $(70, 57)$ m. The following cost function aims at proceeding towards the goal, while minimizing the applied thrust:

$$\min_{\boldsymbol{p}(k),\boldsymbol{u}(k)} J_T = \sum_{k=0}^{T-1} \left(\boldsymbol{q}'|\boldsymbol{p}(k) - \boldsymbol{p}_f| + \boldsymbol{r}'|\boldsymbol{u}(k)| \right) + \boldsymbol{s}'|\boldsymbol{p}(T) - \boldsymbol{p}_f| \tag{2.36}$$

Here $\boldsymbol{p}(k)$ denotes the position $(x(k), y(k))$ of the aircraft, and $\boldsymbol{q}, \boldsymbol{r}$ and $\boldsymbol{s}$ are appropriate weighting vectors. The absolute values in the cost function can be handled by introducing auxiliary variables and additional constraints according to the following principle. Using an auxiliary variable z, the problem $\min |x|$ is equivalent to (see [18]):

$$\begin{aligned} &\min z \\ &\text{s.t.} \quad x \leq z \\ &\qquad -x \leq z. \end{aligned} \tag{2.37}$$

2.4.2 Example 2: Multiple UAVs in 2D

The second scenario, plotted in Figure 2-4, involves three aircraft ($V = 3$) flying at 150 m/s that are initially positioned along a circle. They have to move to the opposite sides (e.g., because their individual reference trajectories are straight lines through the origin) and will therefore cross in the middle. All aircraft are identical with the following parameters: $T = 10$ with $\Delta t = 5$ s, $v_{max} = 160$ m/s, $v_{max} = 130$ m/s, $\omega_{max} = 5$ deg/s, $d_{safe} = 2$ km,

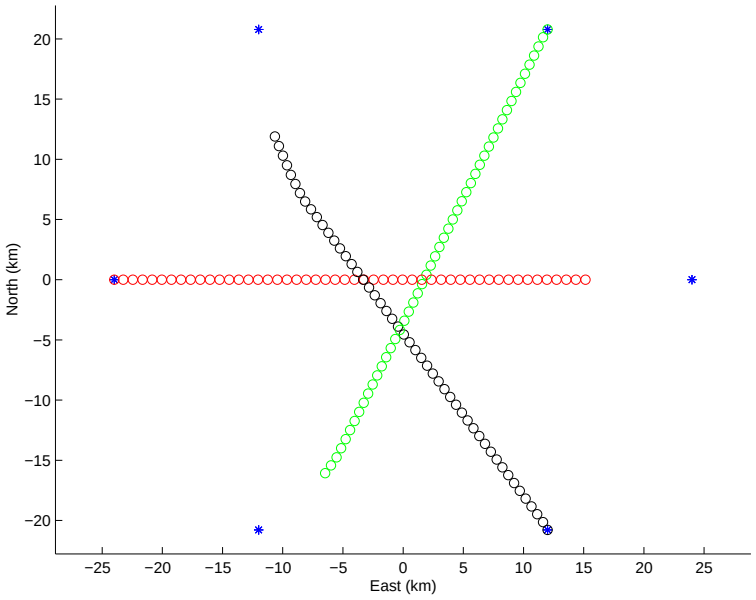


Figure 2-4: Example 2: The aircraft are initially positioned along a circle and need to fly to the opposite side. A conflict in the middle is avoided thanks to the collision avoidance constraints.

and $N = 8$. A multi-vehicle version of cost function (2.36) was used:

$$\min_{\boldsymbol{p}(k),\boldsymbol{u}(k)} J_V = \sum_{i=1}^{V} \sum_{k=0}^{T-1} \left(\boldsymbol{q}'|\boldsymbol{p}_i(k) - \boldsymbol{p}_{f,i}| + \boldsymbol{r}'|\boldsymbol{u}_i(k)| \right) + \boldsymbol{s}'|\boldsymbol{p}_i(T) - \boldsymbol{p}_{f,i}| \tag{2.38}$$

where the weight vectors $\boldsymbol{q}$ and $\boldsymbol{r}$ where set to $\mathbf{1}$ and $\boldsymbol{s}$ to $\mathbf{100}$. The trajectories clearly indicate the effect of the collision avoidance constraints: two of the aircraft change their heading in order to maintain a safe distance near the origin.

2.4.3 Example 3: Helicopter in 3D

In the last example, shown in Figures 2-5 and 2-6, a helicopter with $v_{max} = 20$ m/s, $a_{max} = 3$ m/s^2, $\dot{z}_{max} = |\dot{z}_{min}| = 4$ m/s and $\ddot{z}_{max} = |\ddot{z}_{min}| = 2$ m/s^2 has to fly through a 3D urban environment. It starts in from hover at location $(30, -60, 8)$ m and must fly to hover in $(125, 60, 10)$ m. The planning horizon consists of $T = 10$ time steps of 1 s each, thus resulting in a 20 m separation between trajectory points if flying at the maximum speed. Since this would give large safety envelopes compared to the size of the obstacles, 4 interpolation points are used ($L_a = 4$).

The cost function used in this example automatically switches from a 1-norm minimizing one to a minimum time objective once the goal is within reach of the planning horizon. This is done as follows:

$$J_T^* = \min \sum_{k=1}^{T} \boldsymbol{q}'|\boldsymbol{x}(k) - \boldsymbol{x}_f| + \sum_{k=0}^{T-1} \boldsymbol{r}'|\boldsymbol{u}(k)| + \sum_{k=1}^{T} M_t(T-k)\hat{t}(k) \tag{2.39}$$

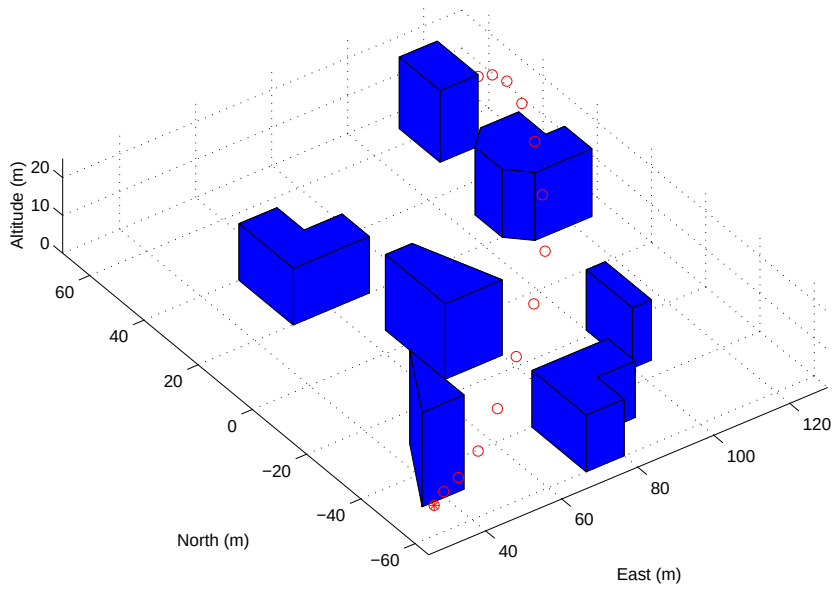


Figure 2-5: Example 3: The helicopter is initially hovering in position $(30, -60, 8)$ m, and has to maneuver to hover in position $(125, 60, 10)$ m. The goal is reached after 14 s.

subject to:

$$\boldsymbol{x}(k) - \boldsymbol{x}_f \leq M_t \hat{t}(k),\ k = 1, \ldots, T \tag{2.40}$$

$$\boldsymbol{x}_f - \boldsymbol{x}(k) \leq M_t \hat{t}(k),\ k = 1, \ldots, T \tag{2.41}$$

where $\hat{t}(k)$ are binary variables and M_t is a sufficiently large number again. However, this time that number should also be greater than any of the values the 1-norm terms in the cost function can take. In that case, if the goal is within reach, the optimal solution will have as many binaries $\hat{t}(k)$ as possible set to 0, enforcing the vehicle to reach the goal state $\boldsymbol{x}_f$ as quickly as possible. In the example, the desired hover position is reached after 14 s. If the final speed was not constrained, the helicopter would fly through the point at maximum speed, eventually trying to return to it to further minimize the cost. In the example, M_t was set to 10000 and $\boldsymbol{q}$ and $\boldsymbol{r}$ were set respectively to $\mathbf{1}$ and $\mathbf{10^{-3}}$.

Finally, it is worthwhile to mention that the preceding scenarios were computed in real-time, i.e., all iterations terminated well within the time step duration of 1 s in examples 1 and 3, and within 5 s in example 2. The computations were done using MATLAB and CPLEX 8.1 on a Pentium 4 PC with 2 GHz clock speed.

2.5 Conclusion

This chapter presented the mathematical problem formulation of the trajectory planning problem for single and multiple vehicles using a receding horizon optimization strategy. Mixed-integer linear programming was used as the implementation framework and constraints ensuring dynamic feasibility, avoidance of arbitrarily shaped obstacles and of collisions were developed in detail. Three scenarios were presented, illustrating the generality and flexibility of the MILP approach and introducing various cost functions.

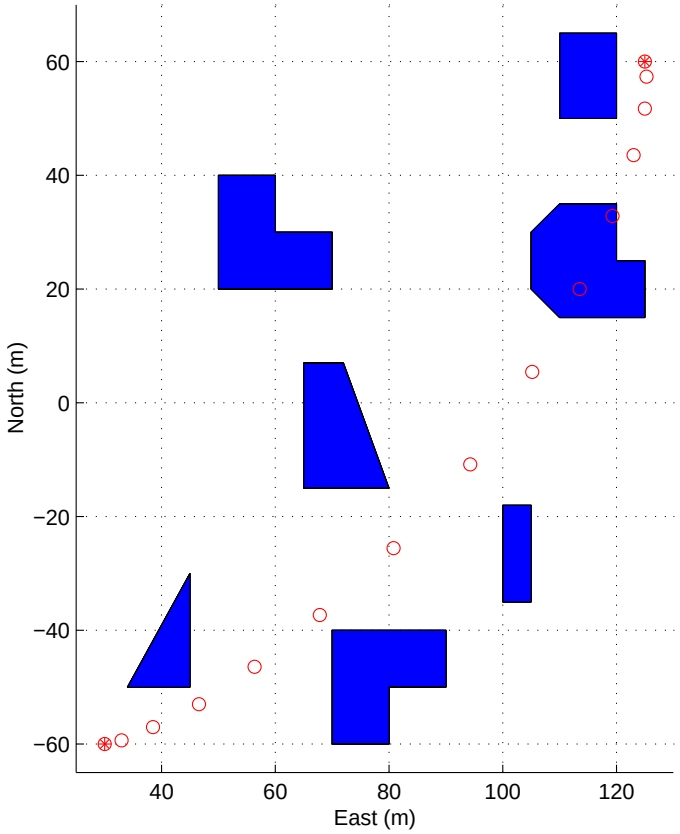


Figure 2-6: Two-dimensional projection of Example 3

Chapter 3

Hybrid Model for Agile Vehicles

This chapter presents a hybrid model and control architecture for trajectory planning of agile vehicles. A velocity control system providing the ability to accurately track trajectories is combined with a maneuver scheduler that enables execution of pre-programmed agile maneuvers. The closed-loop dynamics under this control architecture are described by a simple hybrid model, consisting of a set of constrained, linear time-invariant modes and discrete fixed-duration transitions in the state space. Given these dynamics, mixed-integer linear programming is again used to compute optimal trajectories in cluttered environments. Continuous constraints and binary logic are combined to model the constraints governing the dynamics, to formulate switching rules between different velocity modes, to encode execution of agile maneuvers, and to account for obstacle avoidance. Both offline time-optimal planning and online receding horizon formulations are presented. The framework is applied in detail to a small-scale helicopter, for which several receding horizon results are given. A discussion about practical considerations regarding a real-time implementation concludes the chapter.

3.1 Introduction

Taking advantage of the full range of vehicle maneuverability in an autonomous fashion is key to a number of potential UAV tasks. Examples include tracking of moving targets, flying through cluttered (e.g., urban) environments, and tactical flight, such as nap of the earth. In these applications, autonomous agile maneuvering can either be necessary – for instance to effectively avoid pop-up obstacles– or may represent a competitive advantage. However, as discussed earlier, it is known that motion planning is intrinsically PSPACE-hard [110, 49] and that its complexity grows with that of the vehicle dynamics and the environment in which the vehicle has to operate. For vehicles with fast and complex dynamics, such as small-scale rotorcraft [43], it is therefore impractical to consider the full equations of motion in the development of an autonomous trajectory planning system. For such vehicles, the state space and the set of possible control actions are extremely large, requiring simplifications to reduce the dimensionality when a solution has to be computed in real-time.

A well-established idea to reduce the complexity is to first organize the vehicle dynamics through some form of control augmentation, such as velocity controllers, and to then design the guidance system using the simpler closed-loop dynamics. However, for highly agile vehicles, such an approach might restrict the performance. An alternative method is to use a "maneuver automaton" as was introduced by Frazzoli et al. [36]. The framework

was later applied to an X-Cell miniature helicopter as described in [93] and [128]. With this approach, the vehicle is modeled as a hybrid automaton, consisting of a set of discrete equilibrium trim conditions and transitions between these trims, called maneuvers. By choosing an appropriate set of such motion primitives, the state space can be significantly reduced without giving up the key performance and maneuverability of the vehicle. Optimal trajectories are obtained in real-time by evaluating the possible discrete actions at each time step, i.e., to stay on the current trim trajectory or to execute a maneuver. The decision is made according to an optimal policy operating on a value function, which results from a dynamic program that is solved offline by value iteration [17].

The maneuver automaton as described above, however, has several drawbacks. These are primarily related to the fact that the vehicle dynamics are constrained to a finite set of motion primitives. Namely, velocity is discretized into several trims with constant speeds, thus restricting the vehicle's behavior to one of these. The lack of continuous velocity modes and the discretization used in the value function can be a problem when precise navigation is required, as illustrated by our results in [93, 128]. Moreover, since one operating region is typically discretized into multiple trim conditions with corresponding transition maneuvers, the complexity of the maneuver automaton and corresponding dynamic program increases significantly with the resolution of the discretization.

This chapter presents an alternative approach based on a hybrid architecture that combines a velocity control system and a maneuver scheduler. Using this framework, optimal trajectory design can again be formulated as a mixed-integer linear program. Besides allowing for obstacle and collision avoidance, in this specific case, MILP is used to optimally switch between various velocity control modes and to incorporate the binary decisions whether or not to execute a maneuver. As such, this chapter extends the basic trajectory planning formulation presented in Chapter 2 by accounting for more agile dynamics than the ones used before. We will specialize the approach to the case of a small-scale rotorcraft using a MILP model based on MIT's aerobatic X-Cell helicopter.

The chapter is organized as follows. Section 3.2 presents the hybrid control architecture and a corresponding high-level dynamic model. Given these dynamics, Section 3.3 outlines the use of MILP for optimal guidance. In Section 3.4, the framework is applied to guidance of a small-scale helicopter, for which simulation results are presented in Section 3.5. Section 3.6 then discusses some practical considerations regarding a real-time implementation.

3.2 Hybrid Control Architecture for Guidance

3.2.1 Automatic Control of Agile Vehicles

The control architecture presented in this chapter is based on the analysis of human control of highly agile, small-scale helicopters [40]. It shows two distinct operating regimes: tracking of trim trajectories and maneuvering. The two modes are distinctly set apart in terms of control strategy and dynamic conditions:

- Tracking operations take place around trim trajectories. Control around these trajectories involves continuous feedback, and the dynamics are approximately linear.

- Maneuvering actions are of finite duration and start and end on trim trajectories. The control activity typically involves large amplitude input commands that exploit the extreme performance of the vehicle and result in large changes of its state. The

dynamics across this range are typically nonlinear. Control is dominated by feedforward actions; feedback may include discrete switching events triggered by state thresholds.

Tracking trim trajectories is a well-researched area; automatic maneuvering, however, is more challenging due the highly nonlinear dynamics [48]. Instead of applying traditional nonlinear control methods such as feedback linearization [140], a control logic inspired by the above human strategies was recently developed for an autonomous miniature helicopter [44]. It combines angular rate controllers and a timing logic that enables tracking of pre-programmed reference trajectories. The amplitude and timing of these trajectories or maneuvers were extracted from piloted flight-test experiments. Prior to and upon exit from a maneuver, gain-scheduled linear quadratic trim tracking controllers are used. Using this approach, several aerobatic maneuvers were successfully implemented, including a snap roll, a hammerhead, and a split-S [41].

A block diagram of the switching control architecture is shown in Figure 3-1. The velocity controllers enable the vehicle to accurately track trajectories throughout a large region of the flight envelope, whereas the maneuvers take the helicopter through its extreme range of performance. Under this control architecture, the closed-loop dynamics of the agile vehicle can be accurately described by a combination of low-order, linear time-invariant (LTI) equations of motion and discrete state transitions. An abstract representation of this structure is given in Figure 3-2: it shows that maneuvers start and end in the linear velocity control regime.

More generally, for any type of unmanned agile vehicle, the combination of gain-scheduled LTI modes and a finite number of fast, pre-programmed maneuvers enables a broad range of behaviors that can be exploited when designing mission-specific trajectories. The benefits of this approach are several. First, the model allows for precise navigation in the velocity control mode, without compromising on agility when extreme transitions are required, such as during reactive threat or obstacle avoidance. Second, compared to the maneuver automaton approach [36], the architecture significantly simplifies the development of a motion primitive library: the problem is reduced to selecting a few LTI modes and a small set of agile maneuvers. In what follows, we formalize this control structure in a mathematical framework, which can then be used to effectively formulate trajectory optimization problems.

3.2.2 Velocity Control System

Assume that the gain-scheduled velocity controllers result in L distinct, mutually exclusive LTI modes or operating regions. In general, each LTI mode l corresponds to a discrete-time state space model $\boldsymbol{x}(k+1) = \boldsymbol{A}_l\boldsymbol{x}(k) + \boldsymbol{B}_l\boldsymbol{u}(k)$ in some vector space $\mathbb{R}^{n_s}$, where the index k again indicates the discrete time step. As was discussed in Chapter 2, the state vector $\boldsymbol{x} \in \mathbb{R}^{n_s}$ typically contains velocity and position components in either a body-fixed and/or inertial coordinate frame. The input vector $\boldsymbol{u} \in \mathbb{R}^{n_u}$ generally consists of reference velocity commands or accelerations, again in either coordinate frame. The key requirement is that the state space models describe the closed-loop dynamics of the vehicle in a form that is compatible with the type of desired guidance. For example, if a waypoint follower is available, a closed-loop model incorporating this controller could be considered that takes inertial positions as inputs. Alternatively, it may be desirable to directly steer the vehicle using body-fixed frame velocity commands. Since the formulation that we will present can

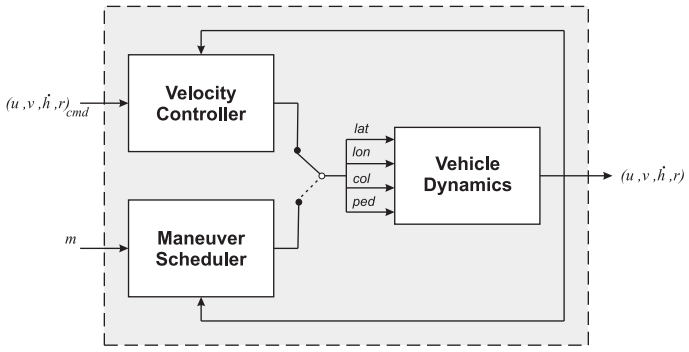


Figure 3-1: Hybrid control architecture: the helicopter can be controlled through a velocity control system, or through a maneuver scheduler that allows the implementation of agile maneuvers taken from a library.

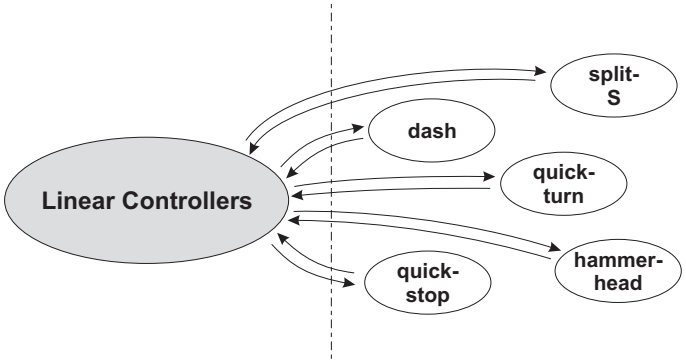


Figure 3-2: Abstract representation of the control architecture. The linear controllers allow continuous motion, the maneuvers are discrete transitions.

capture both coordinate frames or a combination thereof, we will make abstraction of this distinction. Using the state space model, the solution to the trajectory planning algorithm will consist of a sequence of inputs in the appropriate form.

Besides the state space matrices $(\boldsymbol{A}_l, \boldsymbol{B}_l)$, each operating region l is characterized by a set of feasible states $\mathcal{X}_l \subseteq \mathbb{R}^{n_s}$ and feasible inputs $\mathcal{U}_l \subseteq \mathbb{R}^{n_u}$, such that when $\boldsymbol{x} \in \mathcal{X}_l$, the vehicle is in mode l and only inputs $\boldsymbol{u} \in \mathcal{U}_l$ are allowed. Among other, these constraint sets capture bounds on velocity, acceleration, and turn rate for each LTI mode, as well as obstacle avoidance requirements, which, for brevity of notation, we now include in the state constraint set $\mathcal{X}_l$. The overall constraint sets $\mathcal{X}_l$ and $\mathcal{U}_l$ are typically non-convex (e.g. because of the presence of obstacles in the environment), but subsets of these constraints, such as maximum speed, may be convex. Hence, the sets can always be approximated by a combination of polyhedral regions and intersections of polyhedra, and be described as

follows:

$$\mathcal{X}_l = \begin{cases} \boldsymbol{x}^T \boldsymbol{f}_{l,1}^c \leq 0 & \text{and} \quad \boldsymbol{x}^T \boldsymbol{f}_{l,2}^c \leq 0 \quad \ldots \quad \text{and} \quad \boldsymbol{x}^T \boldsymbol{f}_{l,L_f^c}^c \leq 0 \\ \boldsymbol{x}^T \boldsymbol{f}_{l,1}^n \geq 0 & \text{or} \quad \boldsymbol{x}^T \boldsymbol{f}_{l,2}^n \geq 0 \quad \ldots \quad \text{or} \quad \boldsymbol{x}^T \boldsymbol{f}_{l,L_f^n}^n \geq 0 \end{cases} \tag{3.1}$$

$$\mathcal{U}_l = \begin{cases} \boldsymbol{u}^T \boldsymbol{g}_{l,1}^c \leq 0 & \text{and} \quad \boldsymbol{u}^T \boldsymbol{g}_{l,2}^c \leq 0 \quad \ldots \quad \text{and} \quad \boldsymbol{u}^T \boldsymbol{g}_{l,L_g^c}^c \leq 0 \\ \boldsymbol{u}^T \boldsymbol{g}_{l,1}^n \geq 0 & \text{or} \quad \boldsymbol{u}^T \boldsymbol{g}_{l,2}^n \geq 0 \quad \ldots \quad \text{or} \quad \boldsymbol{u}^T \boldsymbol{g}_{l,L_g^n}^n \geq 0 \end{cases} \tag{3.2}$$

Here, the vectors $\boldsymbol{f}_{l,\cdot}^c$ and $\boldsymbol{f}_{l,\cdot}^n$ denote the coefficients of the linear inequalities that respectively capture the convex and non-convex state constraints associated with the operating region l. Similarly, the vectors $\boldsymbol{g}_{l,\cdot}^c$ and $\boldsymbol{g}_{l,\cdot}^n$ define the convex and nonconvex input constraints.

Summarizing, when moving according to a particular LTI mode l, the dynamics are given by:

$$\begin{cases} \boldsymbol{x}(k+1) &= \boldsymbol{A}_l \boldsymbol{x}(k) + \boldsymbol{B}_l \boldsymbol{u}(k) \\ t(k+1) &= t(k) + \Delta t \\ \boldsymbol{x}(k) &\in \mathcal{X}_l \\ \boldsymbol{u}(k) &\in \mathcal{U}_l \end{cases} \tag{3.3}$$

where we introduced an explicit time evolution equation using the discretization step Δt. After applying the control input $\boldsymbol{u}(k)$, the next state $\boldsymbol{x}(k+1)$ can lie in the same operating region l or the vehicle may have transitioned to another mode l'.

3.2.3 Maneuver Scheduler

As mentioned before, the maneuver scheduler allows the execution of pre-programmed maneuvers that can result in rapid and extreme changes of the vehicle's state. Consider a library of P individually designed maneuvers with fixed durations ΔT_m and fixed spatial displacements with respect to the vehicle's body frame. For trajectory planning purposes, each maneuver m can then be characterized by a discrete state update equation in the appropriate reference system, as follows:

$$\begin{cases} \boldsymbol{x}(k+1) &= \boldsymbol{C}_m \boldsymbol{x}(k) + \boldsymbol{d}_m \\ t(k+1) &= t(k) + \Delta T_m \\ \boldsymbol{x}(k) &\in \mathcal{X}_m \end{cases} \tag{3.4}$$

Here, $\boldsymbol{C}_m$ is a fixed matrix and $\boldsymbol{d}_m$ a fixed vector describing the maneuver as an affine transformation of the feasible ingress state $\boldsymbol{x}(k) \in \mathcal{X}_m$. Namely, a particular maneuver m can only be initiated when the state lies within certain bounds described by $\mathcal{X}_m$. For example, a pre-programmed hammerhead maneuver can only be executed above a certain speed. In general, the feasible entry conditions can again be approximated by a set of linear inequalities, capturing both convex and nonconvex constraints on the entry state:

$$\mathcal{X}_m = \begin{cases} \boldsymbol{x}^T \boldsymbol{f}_{m,1}^c \leq 0 & \text{and} \quad \boldsymbol{x}^T \boldsymbol{f}_{m,2}^c \leq 0 \quad \ldots \quad \text{and} \quad \boldsymbol{x}^T \boldsymbol{f}_{m,M_f^c}^c \leq 0 \\ \boldsymbol{x}^T \boldsymbol{f}_{m,1}^n \geq 0 & \text{or} \quad \boldsymbol{x}^T \boldsymbol{f}_{m,2}^n \geq 0 \quad \ldots \quad \text{or} \quad \boldsymbol{x}^T \boldsymbol{f}_{m,M_f^n}^n \geq 0 \end{cases} \tag{3.5}$$

Here, the vectors $\boldsymbol{f}_{m,\cdot}^c$ and $\boldsymbol{f}_{m,\cdot}^n$ again denote the coefficients of the linear inequalities corresponding to the convex and nonconvex constraints associated with maneuver m. The entry state $\boldsymbol{x}(k)$ and exit state $\boldsymbol{x}(k+1)$ must lie in the operating region of some LTI mode, which can be identical, adjacent, or non-adjacent in case of a large jump in the state space.

Typically, the ingress constraint set $\mathcal{X}_m$ is a subset of only one LTI operating region $\mathcal{X}_l$.

Notice that we did not consider a set of feasible control inputs corresponding to a maneuver. Since it is pre-programmed, the only relevant control input associated with a maneuver is the binary decision whether to initiate it or not. This decision is ultimately taken by the trajectory optimization algorithm. Since any details about the physical control variables will be hidden to the optimization problem, the maneuver can be abstracted as a state transition without input.

3.2.4 LTI-Maneuver Automaton

With the corresponding operating and entry constraints, the closed-loop dynamics consisting of the combination of LTI modes and finite duration maneuvers constitute a hybrid input/output automaton [84], which we will call an LTI-maneuver automaton (LTI-MA). A graph representation of this LTI-MA is given in Figure 3-3. During each time step, the vehicle is either in an LTI mode or executing a maneuver. State transitions between two LTI modes, i.e., those described by equations (3.3), take a regular time interval Δt; transitions resulting from performing a particular maneuver m have a duration ΔT_m.

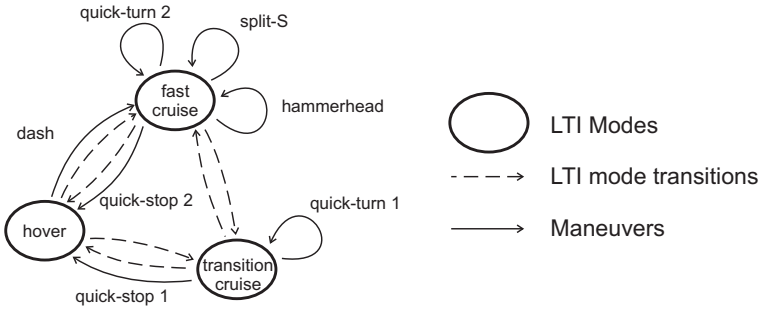


Figure 3-3: Graph representation of the LTI-maneuver automaton. The vehicle is either in an LTI mode, or executing a finite duration maneuver.

3.3 Trajectory Optimization with the LTI-MA

3.3.1 Sequential decision process

Given the LTI-MA framework described above, our goal is to compute optimal trajectories between two waypoints corresponding to an initial and a final state. An optimal trajectory consists of a sequence of dynamically feasible states and corresponding inputs that satisfy the constraints imposed by the environment, and minimize a certain performance criterion captured by a cost function. The latter can be a measure of time or fuel, or a more complex criterion such as visibility or risk. The waypoints are typically provided by a higher level planning algorithm that optimizes a specific task or mission criterion [89, 130, 6, 113]. We will assume here that such a higher planning level is in place.

Given the LTI-MA architecture, trajectory design can be viewed as a sequential decision process, where at the start of each decision step, the helicopter is flying in one of the LTI modes. The guidance problem then comes down to deciding at each decision step whether to stay in the current LTI mode l, to transition to a neighboring mode l', or to execute a

certain maneuver m. However, the last option is only available if the entry conditions $\mathcal{X}_m$ for that maneuver are satisfied. Furthermore, when the vehicle is in the LTI-regime, the decision steps correspond to normal, discrete time steps. In the maneuver execution mode, on the other hand, the single decision step in which a maneuver is executed, corresponds to a number of time steps equivalent to the actual duration of the maneuver. When optimizing time, this difference should be accounted for in the formulation of the trajectory planning problem.

The guidance decision logic just outlined lends itself well to being formulated as a mixed-integer linear program again. The continuous optimization is now done over the states and inputs that are associated with the various LTI modes. The discrete logic and decisions result from partitioning the state space into the distinct LTI modes and from the option of executing maneuvers when the corresponding entry conditions are satisfied. In what follows, we repeatedly apply the general MILP constraint principles (2.18) and (2.19) to derive the trajectory planning formulation with the LTI-MA dynamics.

3.3.2 Trajectory Optimization Using MILP

Operating Region Bounds

We begin by introducing binary variables to capture the *and/or* logic in the convex and nonconvex constraints (3.1), (3.2), and (3.5), respectively describing the feasible state and input sets of the LTI modes and the feasible entry states of the maneuvers. They can be captured in the matrix form of inequality (2.19). As such, for each LTI mode l, we obtain:

$$\begin{cases} \boldsymbol{x}(k+1) &= \boldsymbol{A}_l\boldsymbol{x}(k) + \boldsymbol{B}_l\boldsymbol{u}(k) \\ t(k+1) &= t(k) + \Delta t \\ \mathbf{0} &\geq \boldsymbol{F}_l^c\boldsymbol{x}(k) + \boldsymbol{F}_l^b\boldsymbol{y}_l(k) + \boldsymbol{f}_l \\ \mathbf{0} &\geq \boldsymbol{G}_l^c\boldsymbol{u}(k) + \mathbf{G}_l^b\boldsymbol{w}_l(k) + \boldsymbol{g}_l \end{cases} \tag{3.6}$$

where $\boldsymbol{y}_l(k)$ and $\boldsymbol{w}_l(k)$ are binary vectors. The matrix $\boldsymbol{F}_l^c$ combines the coefficients $\boldsymbol{f}_{l,\cdot}^c$ and $\boldsymbol{f}_{l,\cdot}^n$ of the state inequalities (3.1), while matrix $\boldsymbol{F}_l^b$ and vector $\boldsymbol{f}_l$ encode the corresponding nonconvexities using binary logic. Similarly, matrix $\boldsymbol{G}_l^c$ combines the coefficients $\boldsymbol{g}_{l,\cdot}^c$ and $\boldsymbol{g}_{l,\cdot}^n$ of the input inequalities (3.2), and matrix $\boldsymbol{G}_l^b$ and vector $\boldsymbol{g}_l$ encode the associated nonconvexities.

For the maneuver entry conditions (3.5), the same principle yields:

$$\begin{cases} \boldsymbol{x}(k+1) &= \boldsymbol{C}_m\boldsymbol{x}(k) + \boldsymbol{d}_m \\ t(k+1) &= t(k) + \Delta T_m \\ \mathbf{0} &\geq \boldsymbol{F}_m^c\boldsymbol{x}(k) + \boldsymbol{F}_m^b\boldsymbol{z}_m(k) + \boldsymbol{f}_m \end{cases} \tag{3.7}$$

where $\boldsymbol{z}_m(k)$ is again a binary vector, $\boldsymbol{F}_m^c$ combines the coefficients $\boldsymbol{f}_{m,\cdot}^c$ and $\boldsymbol{f}_{m,\cdot}^n$, and $\boldsymbol{F}_m^b$ and $\boldsymbol{f}_m$ encode the nonconvex structure of the entry constraints (3.5).

LTI Mode Switching

At the beginning of each decision step, the vehicle is in exactly one LTI mode l. Hence, with every decision step k, we can associate a binary variable $b_l(k)$ that equals 1 if the vehicle is flying in mode l at the start of that decision step. Since the L modes are mutually exclusive, only one $b_l(k)$ variable out of L can be 1 at each step k. The non-active LTI modes must

then be "switched off". Using principle (2.18), this can be expressed as follows:

$$\forall l \in [1,\ldots,L]: \begin{cases} \boldsymbol{x}(k+1) - \boldsymbol{A}_l\boldsymbol{x}(k) - \boldsymbol{B}_l\boldsymbol{u}(k) & \leq & M(1-b_l(k))\mathbf{1} \\ -\boldsymbol{x}(k+1) + \boldsymbol{A}_l\boldsymbol{x}(k) + \boldsymbol{B}_l\boldsymbol{u}(k) & \leq & M(1-b_l(k))\mathbf{1} \\ \boldsymbol{F}_l^c\boldsymbol{x}(k) + \boldsymbol{F}_l^b\boldsymbol{y}_l(k) \ + \boldsymbol{f}_l & \leq & M(1-b_l(k))\mathbf{1} \\ \boldsymbol{G}_l^c\boldsymbol{u}(k) + \boldsymbol{G}_l^b\boldsymbol{w}_l(k) + \boldsymbol{g}_l & \leq & M(1-b_l(k))\mathbf{1} \end{cases} \tag{3.8a}$$

$$\begin{array}{rcl} t(k+1) - t(k) - \Delta t & \leq & 0 \\ -t(k+1) + t(k) + \Delta t & \leq & 0 \\ \sum_{l=1}^{L} b_l(k) & = & 1 \end{array} \tag{3.8b}$$

where $\mathbf{1}$ represents a unity vector of appropriate length. Notice that the state space equality constraints of the LTI mode dynamics (3.6) have been replaced by a pair of positive and negative inequalities, such that they can be relaxed if the vehicle is not in mode l (i.e., when $b_l(k) = 0$). Since the time evolution equation is the same for all LTI modes, however, the corresponding inequalities need not be relaxed.

Maneuver Execution

Similarly, we introduce a binary selection variable $d_m(k)$ for each maneuver m at each decision step k, which equals 1 if the maneuver is executed:

$$\forall m \in [1,\ldots,P]: \begin{cases} \boldsymbol{x}(k+1) - \boldsymbol{C}_m\boldsymbol{x}(k) - \boldsymbol{d}_m & \leq & M\left(1-d_m(k)\right)\mathbf{1} \\ -\boldsymbol{x}(k+1) + \boldsymbol{C}_m\boldsymbol{x}(k) + \boldsymbol{d}_m & \leq & M\left(1-d_m(k)\right)\mathbf{1} \\ \boldsymbol{F}_m^c\boldsymbol{x}(k) + \boldsymbol{F}_m^b\boldsymbol{z}_m(k) + \boldsymbol{f}_m & \leq & M\left(1-d_m(k)\right)\mathbf{1} \\ t(k+1) - t(k) - \Delta T_m & \leq & M\left(1-d_m(k)\right) \\ -t(k+1) + t(k) + \Delta T_m & \leq & M\left(1-d_m(k)\right) \end{cases} \tag{3.9a}$$

$$\sum_{m=1}^{P} d_m(k) \leq 1 \tag{3.9b}$$

The inequality $\sum_{m=1}^{P} d_m(k) \leq 1$ ensures that at most one maneuver is performed at a time. This implies that if the initial conditions $\boldsymbol{F}_m^c\boldsymbol{x}(k) + \boldsymbol{F}_m^b\boldsymbol{z}_m(k) + \boldsymbol{f}_m \leq \mathbf{0}$ for a certain maneuver m are satisfied, it does not necessarily have to be executed: $d_m(k)$ can still be set to 0. However, if it is initiated, the state- and time update inequalities of the LTI mode constraints (3.8a) and (3.8b) must be relaxed, since they are in that case determined by equations (3.4). We therefore extend constraints (3.8a) and (3.8b) as follows:

$$\forall l \in [1,\ldots,L]: \begin{cases} \boldsymbol{x}(k+1) - \boldsymbol{A}_l\boldsymbol{x}(k) - \boldsymbol{B}_l\boldsymbol{u}(k) & \leq & M(1-b_l(k))\mathbf{1} + M(\sum_{m=1}^{P} d_m(k))\mathbf{1} \\ -\boldsymbol{x}(k+1) + \boldsymbol{A}_l\boldsymbol{x}(k) + \boldsymbol{B}_l\boldsymbol{u}(k) & \leq & M(1-b_l(k))\mathbf{1} + M(\sum_{m=1}^{P} d_m(k))\mathbf{1} \\ \boldsymbol{F}_l^c\boldsymbol{x}(k) + \boldsymbol{F}_l^b\boldsymbol{y}_l(k) \ + \ \boldsymbol{f}_l & \leq & M\left(1-b_l(k)\right)\mathbf{1} \\ \boldsymbol{G}_l^c\boldsymbol{u}(k) + \boldsymbol{G}_l^b\boldsymbol{w}_l(k) + \boldsymbol{g}_l & \leq & M\left(1-b_l(k)\right)\mathbf{1} \end{cases} \tag{3.10a}$$

$$\begin{array}{rcl} t(k+1) - t(k) - \Delta t & \leq & M\sum_{m=1}^{P} d_m(k) \\ -t(k+1) + t(k) + \Delta t & \leq & M\sum_{m=1}^{P} d_m(k) \\ \sum_{l=1}^{L} b_l(k) & = & 1 \end{array} \tag{3.10b}$$

If no maneuver is performed, the extra relaxation term $M(\sum_{m=1}^{P} d_m(k))\mathbf{1}$ will equal $\mathbf{0}$. Since all maneuvers start and end in one of the LTI modes, the LTI operating bounds need

not be relaxed by the maneuver selection variables: even if a maneuver is executed, the vehicle will still be in some LTI mode l at the start of decision step k.

3.3.3 Planning Strategies

Given the preceding LTI-MA dynamics, our next step is to compute an optimal trajectory from an initial state $\boldsymbol{x}_{init}$ to a desired final state $\boldsymbol{x}_f$. As discussed before, these states are typically positions with associated speeds and heading angles provided in an inertial coordinate frame. The desired state $\boldsymbol{x}_f$ will again be provided by a higher level task- or mission design algorithm that we assume is in place. Depending on the operational objective and the available computational resources, we can consider different planning strategies.

Fixed Horizon Planning

A first approach is to solve one (off-line) MILP problem over a fixed number of decision steps, with the constraint that the vehicle must have arrived within some bound of the final state by the end of the planning horizon. This approach requires that all relevant information about the mission and environment is known ahead of time, and that during execution of the plan, the trajectory can be accurately tracked. Introducing a planning horizon of T decision steps and an arrival tolerance $\boldsymbol{\epsilon}$, the optimization problem has the following general form:

$$\min J_{fh} = \sum_{k=0}^{T-1} \left(\ell_k(\boldsymbol{x}(k), \boldsymbol{u}(k), \boldsymbol{b}(k)) + \sum_{l=1}^{L} b_l(k)\ell_{kl} + \sum_{m=1}^{P} d_m(k)\ell_{km} \right) \tag{3.11a}$$

$$\text{subject to} \begin{cases} (3.9a), (3.9b), \ k = 0, \ldots, T-1 \\ (3.10a), (3.10a), \ k = 0, \ldots, T-1 \\ \boldsymbol{x}(T) - \boldsymbol{x}_f \leq \boldsymbol{\epsilon} \\ -\boldsymbol{x}(T) + \boldsymbol{x}_f \leq \boldsymbol{\epsilon} \\ \boldsymbol{x}(0) = \boldsymbol{x}_{init} \end{cases} \tag{3.11b}$$

Here, $\ell_k(\boldsymbol{x}(k), \boldsymbol{u}(k), \boldsymbol{b}(k))$ is a general (piece-wise) linear cost term associated with the k^{th} decision step, with $\boldsymbol{b}(k)$ some binary decision vector. ℓ_{kl} and ℓ_{km} are constant cost factors, respectively associated with being in LTI mode l or executing maneuver m during decision step k. For the problem to be feasible, the horizon length T must be an upper bound on the number of steps needed to reach the desired state.

Shortest Arrival Time

A special case of a fixed horizon planning problem is to compute the shortest time trajectory between the states $\boldsymbol{x}_{init}$ and $\boldsymbol{x}_f$. The shortest time corresponds to the minimum weighted number of decision steps in which the final state can be reached. The weight of each step is the actual duration of the action taken during that step. To minimize the arrival time, we can then introduce binary variables $r(k)$ that select the step at which the final state is reached. In addition, an extra equality constraint is needed that enforces the helicopter to actually reach the desired state at one of the decision steps in the planning horizon [114].

We again consider an horizon of T decision steps and an arrival treshold $\boldsymbol{\epsilon}$. A shortest

time trajectory can then be computed as follows:

$$\min J_{st} = \sum_{k=0}^{T} r(k) k\,\Delta t + \sum_{k=0}^{T-1}\sum_{m=1}^{P} d_m(k)(\Delta T_m - \Delta t) \tag{3.12a}$$

$$\text{subject to } \begin{cases} (3.9a), (3.9b),\ k = 0, \ldots, T-1 \\ (3.10a), (3.10a),\ k = 0, \ldots, T-1 \\ \boldsymbol{x}(k) - \boldsymbol{x}_f \leq \boldsymbol{\epsilon} + M(1 - r(k))\mathbf{1},\ k = 0, \ldots, T \\ -\boldsymbol{x}(k) + \boldsymbol{x}_f \leq \boldsymbol{\epsilon} + M(1 - r(k))\mathbf{1},\ k = 0, \ldots, T \\ \sum_{k=0}^{T} r(k) = 1 \\ \boldsymbol{x}(0) = \boldsymbol{x}_{init} \end{cases} \tag{3.12b}$$

Since only one of the $r(k)$ binary variables equals 1, the first term in the cost function yields the optimal number of decision steps needed to reach the final state, weighted by Δt. If the optimal action at the k^{th} decision step is to stay in the LTI regime, the weight of the step corresponds to the discretization step Δt. However, if the optimal action is to perform a maneuver ($d_m(k) = 1$), the weight of the decision step is the maneuver duration ΔT_m. Since the first term in J_{st} already accounts for a weight Δt, the latter is subtracted from the actual maneuver time ΔT_m.

Receding Horizon Planning

Although it is intrinsic to the motion planning problem as well, a drawback of MILP is that the computation time increases at least polynomially with the number of variables and constraints. Therefore, the fixed horizon approach is mainly suited for offline computation of trajectories. For real-time applications, it can only be applied to relatively small problems, i.e., to problems with a reduced set of LTI modes and maneuvers and/or a limited number of decision steps in which the final state can be reached. Even then, however, offline trajectory planning has several disadvantages. First, once the trajectory has been computed, it does not allow for changes in the vehicle dynamics or for modifications in the environment during execution of the plan. As such, a nominal offline planning strategy is not robust to uncertainties or disturbances. Moreover, all necessary information has to be available beforehand, i.e., before the vehicle starts its mission. This is often impossible and would exclude applications such as reconnaissance or terrain exploration. In addition, it is our intention to use the LTI-MA framework in a real-time guidance loop, where the full range of the vehicle's dynamic capabilities can be exploited in a reactive fashion.

As discussed in Chapter 2, the above limitations can be effectively addressed using a receding horizon planning strategy. The path of the vehicle is then computed iteratively and composed of a sequence of locally (time-)optimal segments. The length T of the planning horizon should be chosen as a function of the available computational resources and the distance over which the environment is fully characterized (e.g, as resulting from onboard sensor information). New information about the environment can be incorporated in the optimization problem at each iteration, thus enabling reactive planning capabilities that are crucial when the environment changes or is explored in real-time.

If the length of the planning horizon is relatively short and the problem complexity low enough to solve the fixed horizon problem (3.11a)-(3.11b) or (3.12a)-(3.12b) in real-time, a first approach to a receding horizon implementation is to solve a fixed horizon problem at each time step. As the vehicle moves closer to the goal, the upper bound T on the

number of required decision steps can then be reduced at each iteration. If the waypoints are relatively far apart, however, and the horizon length is consequently relatively long, this method becomes computationally too expensive for use in real-time.

Therefore, at a certain iteration with current state $\boldsymbol{x}_{init}$, we use the following, more general form for the receding horizon optimization problem with the LTI-MA dynamics over T decision steps:

$$\min J_{rh} = \sum_{k=0}^{T-1} \left(\ell_k(\boldsymbol{x}(k), \boldsymbol{u}(k), \boldsymbol{b}(k)) + \sum_{l=1}^{L} b_l(k)\ell_{kl} + \sum_{m=1}^{P} d_m(k)\ell_{km} \right) + \ell_T(\boldsymbol{x}(T), \boldsymbol{x}_f, \boldsymbol{b}(T)) \tag{3.13a}$$

$$\text{subject to } \begin{cases} (3.9a), (3.9b), & k = 0 \dots T-1 \\ (3.10a), (3.10a), & k = 0 \dots T-1 \\ \boldsymbol{x}(0) = \boldsymbol{x}_{init} \end{cases} \tag{3.13b}$$

Since the final state $\boldsymbol{x}_f$ may not be reachable within T decision steps from the current state $\boldsymbol{x}_{init}$, the arrival constraints are replaced by a terminal cost $\ell_T(\boldsymbol{x}(T), \boldsymbol{x}_f, \boldsymbol{b}(T))$ in the objective function. To be applicable to the MILP framework, the terminal cost must have a piece-wise linear or piece-wise constant form. For a discussion on the role of this cost, we refer the reader back to Chapter 2.

3.4 Small-Scale Helicopter Example

We now apply the mathematical framework from Section 3.3 to a small-scale rotorcraft modeled after MIT's aerobatic X-Cell helicopter [142]. To keep the formulation concise, we present an approximate inertial frame model. A physically more comprehensive approach using body-fixed velocities can be found in [126], in which additional binary logic is introduced to encode the associated nonlinear kinematics.

3.4.1 Helicopter LTI Modes

Continuous-Time Version

The velocity control augmentation system of the X-Cell is described in [43] and features the following command variables: body axis forward velocity u_{cmd} and side velocity v_{cmd}, climb rate $\dot{h}_{cmd}$ and yaw rate r_{cmd}. The yaw rate command is mechanized to work as a turn rate command, both at hover and forward flight. In hover, the helicopter uses tail rotor control to turn on the spot. In forward flight, lateral cyclic and tail rotor control are mixed by the control law to achieve coordinated turns, i.e., turns with zero side-slip. The control system uses gain scheduling to linearize the closed-loop dynamics around several operating velocities, resulting in different LTI modes. As such, the closed-loop dynamics of the X-Cell in each LTI mode l can be accurately modeled by the following decoupled, first-order differential equations [39]:

$$\begin{aligned} \dot{u} &= -\tfrac{1}{\tau_{u_l}} u + \tfrac{1}{\tau_{u_l}} u_{cmd} \\ \dot{v} &= -\tfrac{1}{\tau_{v_l}} v + \tfrac{1}{\tau_{v_l}} v_{cmd} \\ \ddot{h} &= -\tfrac{1}{\tau_{\dot{h}_l}} \dot{h} + \tfrac{1}{\tau_{\dot{h}_l}} \dot{h}_{cmd} \\ \dot{r} &= -\tfrac{1}{\tau_{r_l}} r + \tfrac{1}{\tau_{r_l}} r_{cmd} \end{aligned} \tag{3.14}$$

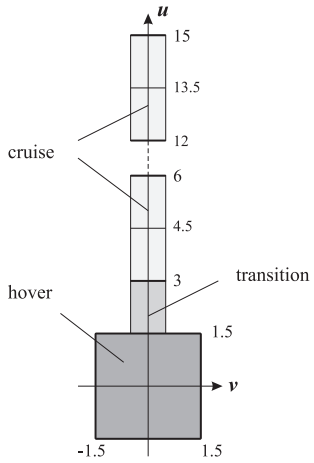


Figure 3-4: Example of operating regions of the velocity control mode, shown in function of the body axis forward (u) and lateral velocity (v).

in which u is the body-fixed forward velocity, v is the body-fixed lateral or side-slip velocity, $\dot{h}$ is the climb rate, and r is the turn rate. Each operating region l is characterized by specific time constants $\tau_{._l}$ and limits on speed, acceleration and control variables. For example, the turn rate response is quicker in hover than in forward flight, and side-slip velocities are not permitted in cruise mode. Figure 3-4 shows a graphical example of this partitioning: the three principal regions are 1) the hover mode, where side-slip and turns on the spot are allowed, 2) the transition mode, where a small amount of side-slip is tolerated, and 3) the cruise mode, where turns are fully coordinated.

Unfortunately, the inertial kinematics associated with this model are nonlinear. We therefore approximate the body-fixed dynamics (3.14) by the following model in an inertial (x, y, z) coordinate frame, in which x and y are the coordinates in the horizontal plane and z is the altitude:

$$\begin{aligned} \ddot{x}(t) &= -\tfrac{1}{\tau_l}\dot{x}(t) + \tfrac{k_l}{\tau_l}\dot{x}_{cmd}(t) \\ \ddot{y}(t) &= -\tfrac{1}{\tau_l}\dot{y}(t) + \tfrac{k_l}{\tau_l}\dot{y}_{cmd}(t) \\ \ddot{z}(t) &= -\tfrac{1}{\tau_z}\dot{z}(t) + \tfrac{k_z}{\tau_z}\dot{z}_{cmd}(t) \end{aligned} \tag{3.15}$$

Each LTI mode l is characterized by a time constant τ_l and gain k_l associated with the planar motion. For the vertical motion, we consider the same time constant τ_z and gain k_z for all modes, indicating that there is only one climb/descent mode which is fully decoupled from the horizontal motion. In state space form $\dot{\boldsymbol{x}}(t) = \boldsymbol{A_{cl}}\boldsymbol{x}(t) + \boldsymbol{B_{cl}}\boldsymbol{u}(t)$, we obtain:

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{z}(t) \\ \ddot{x}(t) \\ \ddot{y}(t) \\ \ddot{z}(t) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -\frac{1}{\tau_l} & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{\tau_l} & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{1}{\tau_z} \end{bmatrix} \begin{bmatrix} x(t) \\ y(t) \\ z(t) \\ \dot{x}(t) \\ \dot{y}(t) \\ \dot{z}(t) \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{k_l}{\tau_l} & 0 & 0 \\ 0 & \frac{k_l}{\tau_l} & 0 \\ 0 & 0 & \frac{k_z}{\tau_z} \end{bmatrix} \begin{bmatrix} \dot{x}_{cmd}(t) \\ \dot{y}_{cmd}(t) \\ \dot{z}_{cmd}(t) \end{bmatrix} \tag{3.16}$$

The input vector $\boldsymbol{u}$ now contains inertial reference speeds $\dot{x}_{cmd}$, $\dot{y}_{cmd}$, and $\dot{z}_{cmd}$, instead of body-fixed velocities; the state vector $\boldsymbol{x}$ is composed of the inertial position vector $[x\ y\ z]'$

and velocity vector $[\dot{x}\ \dot{y}\ \dot{z}]'$. Accordingly, after discretizing time, the control sequence corresponding to a trajectory will consist of inertial reference velocities that can be transformed into equivalent body-fixed velocity commands. Alternatively, the resulting inertial positions can be given as inputs to a waypoint tracking controller.

Notice, however, that the planar dynamics are isotropic in both coordinates and ignore limits on side-slip and turn rate. To correct for this and distinguish between the different operating regions, we introduce additional constraints $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{u} \in \mathcal{U}$ on the state and input vectors respectively. First, the different LTI modes are scheduled with the magnitude of the planar inertial velocity vector $\boldsymbol{v} = [\dot{x}\ \dot{y}]'$. Each operating region l is thus delimited by a minimum speed $v_{min,l}$ and a maximum speed $v_{max,l}$:

$$v_{min,l} \leq ||\boldsymbol{v}|| \leq v_{max,l} \tag{3.17}$$

where for the hover mode ($l = 1$), we have $v_{min,1} = 0$. The magnitude of the horizontal velocity command vector $\boldsymbol{v}_{cmd} = [\dot{x}_{cmd}\ \dot{y}_{cmd}]'$, however, is the same for all modes:

$$0 \leq ||\boldsymbol{v}_{cmd}|| \leq v_{max} \tag{3.18}$$

where v_{max} is the overall maximum velocity of the helicopter. Similarly, the bounds on the climb rate command $\dot{z}_{cmd}$ and the resulting acceleration $\ddot{z}$ are identical for all LTI modes:

$$\dot{z}_{min} \leq \dot{z}_{cmd} \leq \dot{z}_{max} \tag{3.19a}$$

$$\ddot{z}_{min} \leq \ddot{z} \leq \ddot{z}_{max} \tag{3.19b}$$

where $\dot{z}_{min}$ and $\ddot{z}_{min}$ are negative numbers corresponding to descent.

Second, each LTI mode is characterized by specific bounds on turn rate and side slip. In hover mode, quick turns are possible and the limits on forward and lateral acceleration are of similar magnitude. In cruise mode, however, the helicopter flies in an airplane-like fashion in which no side-slip is permitted and the achievable turn rate is much lower. The latter is then inversely proportional to the forward velocity. Both slide-slip and turn rate constraints, however, can be captured by one mode-dependent geometric profile in which the horizontal inertial acceleration vector $\boldsymbol{a} = [\ddot{x}\ \ \ddot{y}]'$ must lie.

A reasonably good approximation is to consider elliptical areas that – in the most general case– are delimited along their principal axes by mode-dependent maximum forward and lateral accelerations ($a_{fwd,max,l}$ and $a_{lat,max,l}$). For the hover mode, this ellipse will be approximately circular. In cruise, however, $a_{fwd,max,l}$ is typically larger than $a_{lat,max,l}$, corresponding to the fact that a helicopter can typically accelerate or decelerate faster than it can turn. This results in a long and narrow ellipse. Assuming coordinated turns, the ellipse should at all times be aligned with the planar velocity vector $\boldsymbol{v}$, such that it tracks changes in heading. This can be achieved by approximating the ellipse as the intersection of two circles whose centers lie along the line that goes through the origin and is parallel to the orthogonal complement $\boldsymbol{v}^{\perp} = [-\dot{y}\ \dot{x}]'$ of $\boldsymbol{v}$. The geometric construction is illustrated in Figure 3-5 for the cruise flight case $a_{fwd,max,l} \geq a_{lat,max,l}$. Using appropriate parameters α_l and β_l for each LTI mode l, these circles can be formulated as:

$$||\boldsymbol{a} - \alpha_l \frac{\boldsymbol{v}^{\perp}}{||\boldsymbol{v}||}|| \leq \beta_l \tag{3.20a}$$

$$||\boldsymbol{a} + \alpha_l \frac{\boldsymbol{v}^{\perp}}{||\boldsymbol{v}||}|| \leq \beta_l \tag{3.20b}$$

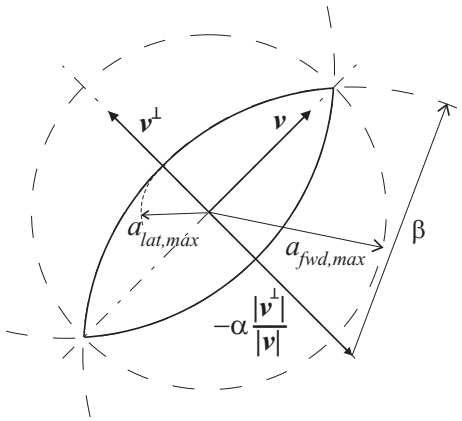


Figure 3-5: Two circle approximation of the elliptic constraint on forward and lateral acceleration.

where – using basic geometry,– α_l and β_l are determined by the values of $a_{fwd,max,l}$ and $a_{lat,max,l}$ as follows:

$$\alpha_l = \frac{a_{fwd,max,l}^2 - a_{lat,max,l}^2}{2a_{lat,max,l}} \tag{3.21a}$$

$$\beta_l = \sqrt{\left(\frac{a_{fwd,max,l}^2 - a_{lat,max,l}^2}{2a_{lat,max,l}}\right)^2 + a_{fwd,max,l}^2} = \sqrt{\alpha^2 + a_{fwd,max,l}^2} \tag{3.21b}$$

In case $a_{fwd,max,l} \leq a_{lat,max,l}$, $\boldsymbol{v}^{\perp}$ should be replaced by $\boldsymbol{v}$ in equalities (3.20a-b) and the role of $a_{fwd,max,l}$ and $a_{lat,max,l}$ be interchanged in equations (3.21a-b). When $a_{fwd,max,l} = a_{lat,max,l}$, the previous expressions result in a circular acceleration profile.

Discrete-Time Version

To make use of the above dynamics in our MILP framework, the state space models (3.16) need to be discretized. Using the bilinear transform with a sample time Δt, the discrete state space model for each LTI mode l becomes: $\boldsymbol{x}(k+1) = \boldsymbol{A}_l\boldsymbol{x}(k) + \boldsymbol{B}_l\boldsymbol{u}(k)$, with

$$\begin{aligned} \boldsymbol{A}_l &= \left(\boldsymbol{I} - \tfrac{\Delta t}{2}\boldsymbol{A_{cl}}\right)^{-1}\left(\boldsymbol{I} + \tfrac{\Delta t}{2}\boldsymbol{A_{cl}}\right) \\ \boldsymbol{B}_l &= \left(\boldsymbol{I} - \tfrac{\Delta t}{2}\boldsymbol{A_{cl}}\right)^{-1}\boldsymbol{B_{cl}}. \end{aligned} \tag{3.22}$$

Next, as depicted in Figure 3-6, and similar to the double integrator model from Chapter 2, all nonlinear inequality constraints characterizing each LTI mode l are approximated by N-sided polygons. The upper bounds of the velocity constraints (3.17)-(3.18) then correspond to the vectors lying inside such a polygon, which can be expressed as follows:

$$\forall n \in [1, \ldots, N]: \qquad \dot{x}(k)\sin\left(\frac{2\pi n}{N}\right) + \dot{y}(k)\cos\left(\frac{2\pi n}{N}\right) \leq v_{max,l} \tag{3.23}$$

$$\dot{x}_{cmd}(k)\sin\left(\frac{2\pi n}{N}\right) + \dot{y}_{cmd}(k)\cos\left(\frac{2\pi n}{N}\right) \leq v_{max} \tag{3.24}$$

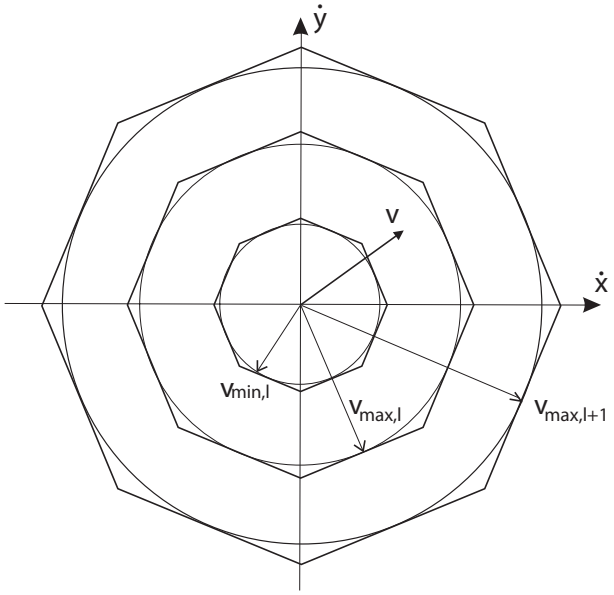


Figure 3-6: Polygonal approximation of minimum and maximum velocity bounds delimiting the various LTI-modes.

As in (2.23)-(2.24), the minimum velocity $v_{min,l}$ delimiting mode l from below in constraint (3.17) can be handled by ensuring that the speed vector lies *outside* the corresponding polygon. By introducing N binary variables $c_{ln}(k)$ and applying principle (2.18), we can encode this nonconvex constraint as follows:

$$\forall n \in [1,\ldots,N]: \quad \dot{x}(k)\sin\left(\frac{2\pi n}{N}\right) + \dot{y}(k)\cos\left(\frac{2\pi n}{N}\right) \;\geq\; v_{min,l} - Mc_{ln}(k) \quad (3.25a)$$

$$\sum_{n=1}^{N} c_{ln}(k) \;\leq\; N-1 \quad (3.25b)$$

where M is again a sufficiently large number.

For the climb and descent acceleration constraints (3.19b), we use an Euler discretization of the derivative:

$$\ddot{z}_{min} \leq \frac{\dot{z}(k+1)-\dot{z}(k)}{\Delta t} \leq \ddot{z}_{max} \quad (3.26)$$

Applying the same principle and the polygonal approximation to the planar inertial constraints (3.20a-b), we obtain:

$\forall n \in [1,\ldots,N]:$

$$\left(\frac{\dot{x}(k+1)-\dot{x}(k)}{\Delta t} + \alpha_l\frac{\dot{y}(k)}{v(0)}\right)\sin\left(\frac{2\pi n}{N}\right) + \left(\frac{\dot{y}(k+1)-\dot{y}(k)}{\Delta t} - \alpha_l\frac{\dot{x}(k)}{v(0)}\right)\cos\left(\frac{2\pi n}{N}\right) \;\leq\; \beta_l \quad (3.27a)$$

$$\left(\frac{\dot{x}(k+1)-\dot{x}(k)}{\Delta t} - \alpha_l\frac{\dot{y}(k)}{v(0)}\right)\sin\left(\frac{2\pi n}{N}\right) + \left(\frac{\dot{y}(k+1)-\dot{y}(k)}{\Delta t} + \alpha_l\frac{\dot{x}(k)}{v(0)}\right)\cos\left(\frac{2\pi n}{N}\right) \;\leq\; \beta_l \quad (3.27b)$$

Because these constraints will typically be formulated over a receding planning horizon T, the unknown magnitude of the velocity vector $\boldsymbol{v}(k) = [\dot{x}(k)\ \dot{y}(k)]'$ is replaced by $v(0)$, the *known* horizontal speed at the initial time step in the horizon (except when $v(0) = 0$). As a result, when $||\boldsymbol{v}(k)|| > v(0)$ the circles will lie further away from each other, thus reducing the feasible region and corresponding maximum forward and lateral acceleration. On the other hand, if $||\boldsymbol{v}(k)|| < v(0)$, the circles will move closer together, thereby increasing the allowable accelerations. However, since only the action corresponding to the first decision step will be implemented (at which the acceleration constraints are exact), this under- or overestimation of the available acceleration in the remaining steps will be compensated for at the next iteration. In the fixed arrival time case, $v(0)$ could be replaced by v_l, the forward velocity around which the dynamics are linearized in mode l (except for hover, for which $v_l = 0$).

Still, if this reshaping of the acceleration profile increases the bounds on forward or lateral acceleration by too much, the following magnitude constraint $||\boldsymbol{a}|| \leq \max(a_{fwd,max,l}, a_{lat,max,l})$ can be added:

$\forall n \in [1, \ldots, N]$:

$$\left(\frac{\dot{x}(k+1) - \dot{x}(k)}{\Delta t}\right) \sin\left(\frac{2\pi n}{N}\right) + \left(\frac{\dot{y}(k+1) - \dot{y}(k)}{\Delta t}\right) \cos\left(\frac{2\pi n}{N}\right) \leq \max\left(a_{fwd,max,l}, a_{lat,max,l}\right) \tag{3.28}$$

3.4.2 Helicopter Maneuvers

Maneuvers are designed to exploit the extreme performance and agility of the vehicle: they typically take advantage of the full control input range and result in large state excursions. The availability of such maneuvers plays an essential role in reactive threat and obstacle avoidance. Table 3.4.2 gives an overview of maneuvers that could be designed; Figure 3-7 shows the corresponding trajectories. Both split-S and hammerhead have already been implemented on MIT's helicopter [41, 42]. Other maneuvers that have since been considered include dash, quick-stop (or deceleration) and quick-turn maneuvers. The split-S and hammerhead can be used to quickly reverse the direction of flight; compared to a level-flight U-turn, they are faster and require no lateral displacements. However, both require a minimum entry speed, whereas a U-turn can be performed at any velocity. Also, the split-S results in an altitude drop, while the hammerhead typically ends at the initial or a higher altitude.

Each maneuver m is characterized by its duration ΔT_m, entry and exit speed $v_{in,m}$ and $v_{ex,m}$, resulting spatial displacement $[\Delta x_h\ \Delta y_h\ \Delta z]'_m$ and change in heading $\Delta\psi_m$. As

Maneuver	Usage
Dash to cruise	rapid acceleration from hover to one of the cruise conditions
Quick-stop	rapid transition from cruise to a full stop (hover)
Quick-turn	rapid turn resulting in a pre-determined heading change
Split-S	reversal of the flight direction with altitude loss
Hammerhead	reversal of the flight direction with altitude gain or zero altitude change

Table 3.1: Description of sample maneuvers that could be implemented on a rotorcraft-type vehicle.

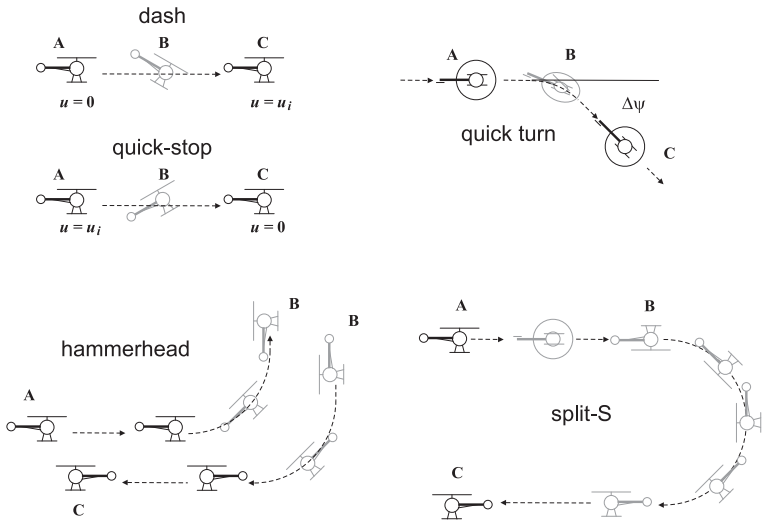


Figure 3-7: Helicopter trajectories for sample maneuvers.

shown in Figure 3-8, the displacement $[\Delta x_h \ \Delta y_h]'_m$ and heading change $\Delta\psi_m$ are defined with respect to the body-fixed frame at the start of the maneuver. From these body frame parameters, a fixed affine transformation of the inertial state vector can be extracted for each maneuver. Assuming that a maneuver can only be initiated in level flight with $\dot{z}_i = 0$ and zero side-slip, the characterizing constants derived from the body frame parameters are:

$$\begin{aligned}
\gamma_m &= -\arctan\left(\frac{\Delta x_{h,m}}{\Delta y_{h,m}}\right) \\
\delta_m &= -\Delta\psi_m \\
c_m &= \frac{\sqrt{(\Delta x_{h,m})^2 + (\Delta y_{h,m})^2}}{v_{init,m}} \\
d_m &= \frac{v_{ex,m}}{v_{in,m}}
\end{aligned}$$

with γ_m defined between $-180°$ and $180°$. The state transition resulting from maneuver m is then given by the following affine transformation:

$$\begin{bmatrix} x(k+1) \\ y(k+1) \\ z(k+1) \\ \dot{x}(k+1) \\ \dot{y}(k+1) \\ \dot{z}(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & c_m\cos\gamma_m & -c_m\sin\gamma_m & 0 \\ 0 & 1 & 0 & c_m\sin\gamma_m & c_m\cos\gamma_m & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & d_m\cos\delta_m & -d_m\sin\delta_m & 0 \\ 0 & 0 & 0 & d_m\sin\delta_m & d_m\cos\delta_m & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x(k) \\ y(k) \\ z(k) \\ \dot{x}(k) \\ \dot{y}(k) \\ \dot{z}(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \Delta z_m \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.29}$$

which combines rotation, scaling and translation to express the exit state $\boldsymbol{x}(k+1)$ as a function of the entry state $\boldsymbol{x}(k)$. The entry conditions $\boldsymbol{x}(k) \in \mathcal{X}_m$ of maneuver m are a fixed planar initial speed $||\boldsymbol{v}(k)|| = v_{in,m}$, zero acceleration $\boldsymbol{a}(k)$, and zero climb rate $\dot{z}(k)$.

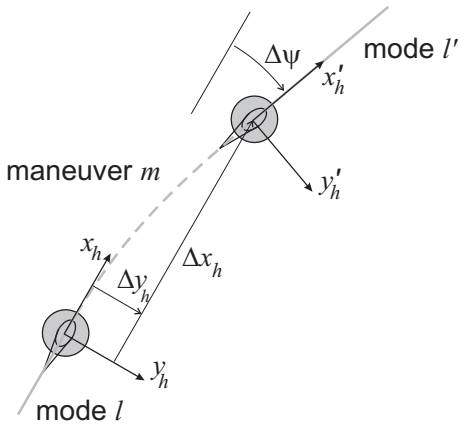


Figure 3-8: Change in helicopter position and orientation resulting from a maneuver (shown in dashed line), as observed from the body-fixed frame

The entry velocity constraint can be expressed as follows:

$$\forall n \in [1,\ldots,N]: \quad \dot{x}(k)\sin\left(\frac{2\pi n}{N}\right) + \dot{y}(k)\cos\left(\frac{2\pi n}{N}\right) \leq v_{in,m} \tag{3.30a}$$

$$\dot{x}(k)\sin\left(\frac{2\pi n}{N}\right) + \dot{y}(k)\cos\left(\frac{2\pi n}{N}\right) \geq v_{in,m} - Mc_{mn}(k) \tag{3.30b}$$

$$\sum_{n=1}^{N} c_{mn}(k) \leq N-1 \tag{3.30c}$$

with $c_{mn}(k)$ binary variables. The acceleration and climb rate conditions become:

$$\begin{aligned} \dot{x}(k) - \dot{x}(k-1) &\leq 0 \\ -\dot{x}(k) + \dot{x}(k-1) &\leq 0 \\ \dot{y}(k) - \dot{y}(k-1) &\leq 0 \\ -\dot{y}(k) + \dot{y}(k-1) &\leq 0 \\ \dot{z}(k) &\leq 0 \\ -\dot{z}(k) &\leq 0 \end{aligned} \tag{3.31}$$

which are relaxed by $M\left(1 - \sum_{m=1}^{P} d_m(k)\right)$ if no maneuver is executed.

3.4.3 Obstacle Avoidance

As was shown in Chapter 2, by constraining the trajectory points to lie outside polyhedral regions, obstacle avoidance can be handled by mixed-integer linear constraints as well. Although any obstacle shape approximated by polyhedrons can be dealt with, for simplicity of exposition, we here only consider rectangular obstacles o that are aligned with the axis frame. We again denote the lower left corners by $(x_{o,min}, y_{o,min}, z_{o,min})$ and upper right corners by $(x_{o,max}, y_{o,max}, z_{o,max})$. As mentioned before, only obstacles that lie within the maximum distance the helicopter can travel over the duration of the planning horizon should be accounted for. Denote the number of these relevant obstacles as S. Then, introducing

binary variables $f_{on}(k)$ for each time step k and each obstacle o, we repeat the avoidance constraints (2.30) as follows:

$$\begin{aligned}
\forall o \in [1,\ldots,S]: \qquad x(k) &\leq x_{o,min} + M f_{o1}(k) \\
y(k) &\leq y_{o,min} + M f_{o2}(k) \\
z(k) &\leq z_{o,min} + M f_{o3}(k) \\
-x(k) &\leq -x_{o,max} + M f_{o4}(k) \\
-y(k) &\leq -y_{o,max} + M f_{o5}(k) \\
-z(k) &\leq -z_{o,max} + M f_{o6}(k) \\
\sum_{e=1}^{6} f_{oe}(k) &\leq 5
\end{aligned} \tag{3.32}$$

Remember that these inequalities encode the requirement that each trajectory point $(x(k), y(k), z(k))$ must lie in at least one of the outer halfspaces defined by the faces of the S obstacles.

Again, because the trajectory consists of a discrete sequence of positions, however, there is no guarantee that the corresponding continuous path does not intersect with the obstacles between two subsequent points, e.g., during a maneuver. The obstacles must therefore be extended by a safety boundary that corresponds to the largest distance that can be traveled during a decision step. If $v_{max}\Delta t$ is large or when maneuvers with big displacements are considered, this safety envelope can be relatively large compared to the size of the obstacles, thus making the obstacle environment denser than might be acceptable. If the vehicle is in the LTI regime, the boundary can be reduced by determining intermediate trajectory points through linear interpolation as in expressions (2.33). For maneuver transitions, an additional avoidance check can be carried out for J sample points along the *actual* maneuver trajectory. Although these points are not known *a priori*, they can be expressed as affine transformations of the position vector $\boldsymbol{p}(k) = [x(k)\ y(k)\ z(k)]'$ and horizontal inertial velocity $\boldsymbol{v}(k) = [\dot{x}(k)\ \dot{y}(k)]'$ at the start of the maneuver. Let $\tilde{\boldsymbol{p}}_{mj}$ be the j^{th} point along the trajectory of maneuver m. Similarly to equation (3.29), by introducing a maneuver-specific constant matrix $\boldsymbol{P}_{mj}$ and constant vector $\boldsymbol{p}_{mj}$ for each sample point $\tilde{\boldsymbol{p}}_{mj}$, we can write:

$$\tilde{\boldsymbol{p}}_{mj} = \boldsymbol{P}_{mj} \begin{bmatrix} \boldsymbol{p} \\ \boldsymbol{v} \end{bmatrix} + \boldsymbol{p}_{mj} \tag{3.33}$$

in which $[\boldsymbol{p}'\ \boldsymbol{v}']'$ is the entry state of the maneuver. At each decision step k, the obstacle avoidance check for maneuver m and obstacle o can then be expressed as follows:

$$\begin{aligned}
\forall o \in [1,\ldots,S], j \in [1,\ldots,J]: \quad \boldsymbol{P}_{mj} \begin{bmatrix} \boldsymbol{p}(k) \\ \boldsymbol{v}(k) \end{bmatrix} + \boldsymbol{p}_{mj} &\leq \begin{bmatrix} x_{o,min} \\ y_{o,min} \\ z_{o,min} \end{bmatrix} + M \begin{bmatrix} f_{jo1}(k) \\ f_{jo2}(k) \\ f_{jo3}(k) \end{bmatrix} \\
-\boldsymbol{P}_{mj} \begin{bmatrix} \boldsymbol{p}(k) \\ \boldsymbol{v}(k) \end{bmatrix} + \boldsymbol{p}_{mj} &\leq -\begin{bmatrix} x_{o,max} \\ y_{o,max} \\ z_{o,max} \end{bmatrix} + M \begin{bmatrix} f_{jo4}(k) \\ f_{jo5}(k) \\ f_{jo6}(k) \end{bmatrix} \\
\sum_{e=1}^{6} f_{joe}(k) &\leq 5
\end{aligned} \tag{3.34}$$

where the same binaries are used for each maneuver m. By adding terms $M(1 - d_m(k))$, the constraints will be relaxed if the maneuver is not executed.

When multiple vehicles are considered, equation (3.33) can also be used to express collision avoidance constraints. By substituting the coordinates in inequalities (2.35) by the sample points (3.33) for each vehicle pair, collision avoidance during execution of maneuvers can be guaranteed. Naturally, the safety distance d_{safe} must then account for the distance traveled between two subsequent maneuver sample points. Because of the large number of binaries that are introduced, however, it is more efficient to tackle collision avoidance in a distributed fashion, as will be discussed in Chapter 5. A particular vehicle then knows *a priori* whether another one is planning to execute a maneuver. As such, that vehicle can consider the corresponding maneuver sample points of all other vehicles as fixed obstacles and use expressions (3.34) accordingly.

3.4.4 Receding Horizon Formulation

Given the above helicopter model, we are interested in guiding the helicopter between waypoints in the fastest possible way, thereby avoiding obstacles. The exact shortest time between two states can be computed using the fixed arrival time approach and objective function (3.12a). However, for real-time applications, we resort to a receding horizon strategy using the following heuristic piece-wise linear cost function. It aims at designing a *fast* trajectory between the initial waypoint $\boldsymbol{p}_{init} = \boldsymbol{p}(0)$ in the horizon and the desired one $\boldsymbol{p}_f$, mimicking a shortest time objective:

$$\min J_h = \sum_{k=0}^{T} -q\boldsymbol{v}'(k)(\boldsymbol{p}_f - \boldsymbol{p}_{init}) + r|\boldsymbol{p}(T) - \boldsymbol{p}_f| \tag{3.35}$$

The first term tries to maximize the scalar product of the inertial velocity with the vector that is pointing from the initial position to the desired one. The effect is twofold: it will speed the helicopter up, while turning it toward the right direction. In addition, the term $r|\boldsymbol{p}(T) - \boldsymbol{p}_f|$ tries to minimize the 1-norm distance towards the goal from the last position in the planning horizon. Both terms thus work towards attaining the same goal, with q and r weight factors emphasizing either effect.

Since the position error term only accounts for the final step of the planning horizon, the helicopter is allowed to speed up away from the goal to enable the execution of a maneuver that will eventually bring it closer than a regular coordinated turn would. This trade-off between acting immediately or "investing" in an overall more efficient maneuver is also captured by the scalar product. For example, consider a case in which the helicopter must reverse direction as quickly as possible, e.g., because of a pop-up threat ahead of it. Assume that it can either make a U-turn or perform a hammerhead. Since the latter requires a minimum entrance speed, depending on the initial velocity, one action may be faster than the other. First speeding up in a straight line to perform the hammerhead will generate scalar product terms that initially increase the objective function, but enable a sudden decrease when the maneuver is executed. Making a U-turn, on the other hand, involves only cost-decreasing terms, but might be slower overall. When there is a bound on lateral displacement, however, e.g., when flying through a street lined with buildings, the nominally fastest action might be infeasible. The trajectory will then be optimized over the alternatives.

Using cost function (3.35), the full MILP problem that must be solved at each receding

horizon iteration becomes:

$$\min J_h = \sum_{k=0}^{T} -q\boldsymbol{v}'(k)(\boldsymbol{p}_f - \boldsymbol{p}(0)) + r|\boldsymbol{p}(T) - \boldsymbol{p}_f|$$

subject to:

$$\begin{cases}
\boldsymbol{x}(k+1) - \mathbf{A}_l\boldsymbol{x}(k) - \mathbf{B}_l\boldsymbol{u}(k) \leq M(1 - b_l(k) + \sum_{m=1}^{P} d_m(k))\mathbf{1} & l = 1,\dots,L,\ k = 0,\dots,T-1 \\
-\boldsymbol{x}(k+1) + \mathbf{A}_l\boldsymbol{x}(k) + \mathbf{B}_l\boldsymbol{u}(k) \leq M(1 - b_l(k) + \sum_{m=1}^{P} d_m(k))\mathbf{1} & l = 1,\dots,L,\ k = 0,\dots,T-1 \\
\boldsymbol{x}(k+1) - \mathbf{C}_m\boldsymbol{x}(k) - \mathbf{d}_m \leq M(1 - d_m(k))\mathbf{1} & m = 1,\dots,P,\ k = 0,\dots,T-1 \\
-\boldsymbol{x}(k+1) + \mathbf{C}_m\boldsymbol{x}(k) + \mathbf{d}_m \leq M(1 - d_m(k))\mathbf{1} & m = 1,\dots,P,\ k = 0,\dots,T-1 \\
\sum_{l=1}^{L} b_l(k) = 1 & k = 0,\dots,T-1 \\
\sum_{m=1}^{P} d_m(k) \leq 1 & k = 0,\dots,T-1 \\
\boldsymbol{x}(0) = \boldsymbol{x}_{init} & \\
\\
(3.19a) & k = 0,\dots,T-1 \\
(3.23) + M(1 - b_l(k)) & l = 1,\dots,L,\ k = 0,\dots,T-1 \\
(3.24) & k = 0,\dots,T-1 \\
(3.25a), (3.25b) + M(1 - b_l(k)) & l = 1,\dots,L,\ k = 0,\dots,T-1 \\
(3.26) & k = 0,\dots,T-1 \\
(3.27a), (3.27b), (3.28) + M(1 - b_l(k)) + M\sum_{m=1}^{P} d_m(k) & l = 1,\dots,L,\ k = 0,\dots,T-1 \\
(3.30a), (3.30b), (3.30c) + M(1 - d_m(k)) & m = 1,\dots,P,\ k = 0,\dots,T-1 \\
(3.31) + M(1 - \sum_{m=1}^{P} d_m(k)) & k = 1,\dots,T-1 \\
(3.32) & k = 1,\dots,T \\
(3.34) + M(1 - d_m(k)) & m = 1,\dots,P,\ k = 0,\dots,T-1
\end{cases} \tag{3.36}$$

3.5 Results

3.5.1 Helicopter and Problem Parameters

We now use the full receding horizon MILP formulation (3.36) to compute real-time reference trajectories for some example scenarios. The helicopter parameters for the LTI modes and maneuvers are based on those of MIT's X-Cell and are given in Tables 3.2 and 3.3. We considered two LTI regimes: 1) a hover mode with forward and lateral velocities up to 3 m/s, and 2) a forward flight cruise mode up to 20 m/s, in which no side-slip is allowed. The maximum forward acceleration for both modes was set to 3 m/s^2, the maximum lateral acceleration for hover and cruise to 3.14 m/s^2 and 3.49 m/s^2 respectively. The latter correspond to maximum turn rates of 60 deg/s and 10 deg/s at the limiting velocities of 3 m/s and 20 m/s.

The maneuver library contains a hammerhead and a split-S with entry speeds of 15 m/s

Mode	v_{min} (m/s)	v_{max} (m/s)	$a_{fwd,max}$ (m/s^2)	$a_{lat,max}$ (m/s^2)	τ (s)	k
Hover	0	3	3	3.14	1	1
Cruise	3	20	3	3.49	1	1

Table 3.2: Parameters of LTI modes

Maneuver	Δx_h(m)	Δy_h(m)	Δz (m)	$\Delta\psi$ (deg)	ΔT(s)	v_{in}(m/s)	v_{ex} (m/s)
Split-S	6.5	10	-30	180	5	15	18
Hammerhead	-20	-6	0	180	7	18	18

Table 3.3: Parameters of pre-programmed maneuvers

and 18 m/s respectively. The body-fixed parameters describing them were obtained by averaging the actual values resulting from autonomous executions of these maneuvers on the X-Cell. Notice that an ideal execution would not exhibit a lateral displacement. However, we chose to keep them because these imperfect maneuvers clearly illustrate the trade-off decisions the MILP optimization makes in the various scenarios. In addition, we focused on 2D trajectory planning and ignored the changes in altitude resulting from a maneuver, thus assuming that the helicopter is flying sufficiently high.

A planning horizon of $T = 5$ decision steps was used with $\Delta t = 1$ s. As such, each iteration of the planning problem had to be solved within a second. Using CPLEX 9.0 on a Pentium 4 with 2.2 GHz clock frequency, optimal or good suboptimal feasible solutions could always be found within this hard real-time limit. The computation times ranged from 0.3 s to the full 1.0 s, but were about 0.6 s on average. The settings of other parameters were as follows: $N = 16$ for the polygonal approximation of circular constraints, $J = 3$ for the obstacle avoidance checks between trajectory points, a 5 m safety boundary around the obstacles, and $q = r = 1$ for the weights in the cost function.

3.5.2 Scenarios

In each of the following scenarios, the helicopter is initially in the origin (0 m, 0 m), flying east at 6 m/s or 12 m/s through a corridor such as a street lined with buildings. It is then given the task to reverse its direction of flight and fly towards location (-100 m, 0 m) as quickly as possible. This command would typically be issued by a higher level decision unit, and could for example result from the detection of a threat ahead of the helicopter.

In the first scenario, depicted in Figure 3-9, the vehicle is flying at 12 m/s. The optimal course of action is to speed up to 15 m/s and execute the split-S, resulting in a quick direction reversal with a 18 m/s exit speed. Conformable to the LTI-MA dynamics, the maneuver is considered as a discontinuous displacement, shown in dashed line. Upon exiting the split-S, the helicopter further accelerates to its maximum speed of 20 m/s. The full sequence to get near the waypoint at (-100 m, 0 m) takes 12 s. Solving once for the exact shortest time trajectory using strategy (3.12a)-(3.12b) yielded the same result. However, the computation took 310 s (for a $T = 14$ step horizon), making the approach inapplicable to online optimization.

In the second scenario, shown in Figure 3-10, the initial speed is 6 m/s. The helicopter now decides to slow down to the hover mode in which it can turn around faster. It now takes 6 s to make the full U-turn and 10 s total to reach the waypoint at a speed of 20 m/s. The exact time-optimal trajectory looked very similar and lasted equally long, but took 471 s to compute (again using 14 decision steps as an upper bound).

Next, we considered two scenarios in which an additional obstacle was placed in the environment. The resulting trajectories are plotted in Figures 3-11 and 3-12, for the two starting velocities of 12 m/s and 6 m/s respectively. Because of the restrictive bound on turn rate in the cruise mode, executing the split-S as in scenario 1 would prohibit the

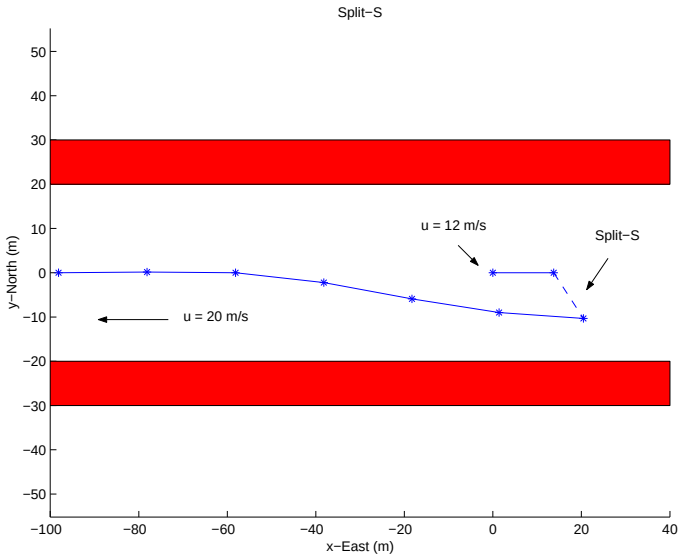


Figure 3-9: Scenario 1: The helicopter is initially in $(0, 0)$ flying at 12 m/s and needs to reverse direction towards $(-100, 0)$ as quickly as possible. It decides to execute the split-S.

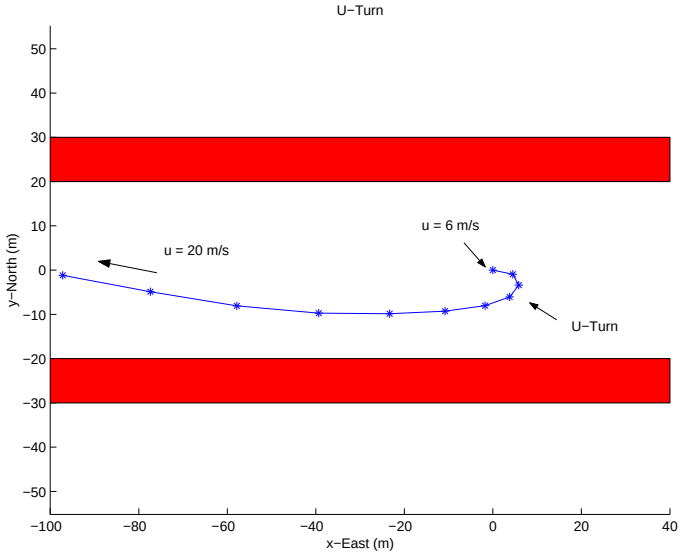


Figure 3-10: Scenario 2: The helicopter is initially in $(0, 0)$ flying at 6 m/s and needs to reverse direction towards $(-100, 0)$ as quickly as possible. It decides to make a U-turn.

vehicle from safely avoiding the obstacle (which is enlarged with a 5 m safety boundary): the helicopter would be facing it with a speed of 18 m/s and insufficient distance to turn away. The trajectory optimization therefore results in the vehicle changing heading during the first time step. That way it is not pointing towards the obstacle after executing the split-S, and has enough space to turn around it. In addition, the helicopter slows down to avoid the southern wall. In the U-turn case of Figure 3-12, the vehicle simply makes a wider coordinated turn.

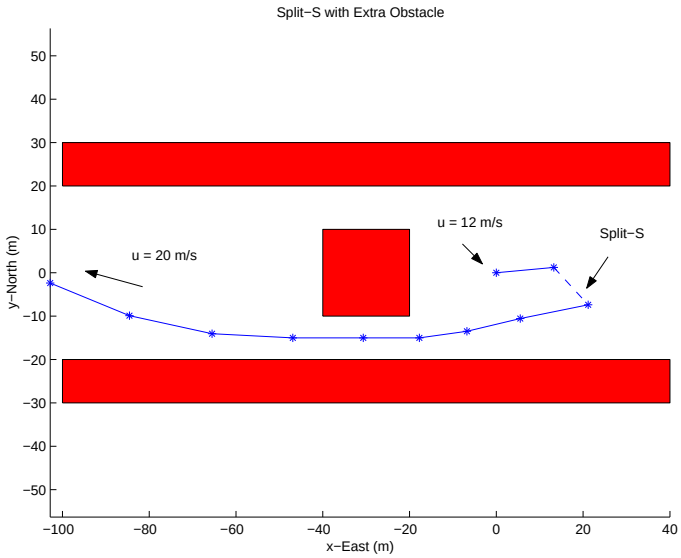


Figure 3-11: Scenario 3: The helicopter is initially in $(0, 0)$ flying at 12 m/s and needs to reverse direction towards $(-100, 0)$ as quickly as possible. To avoid the obstacle in the middle, it decides to change heading before executing the split-S.

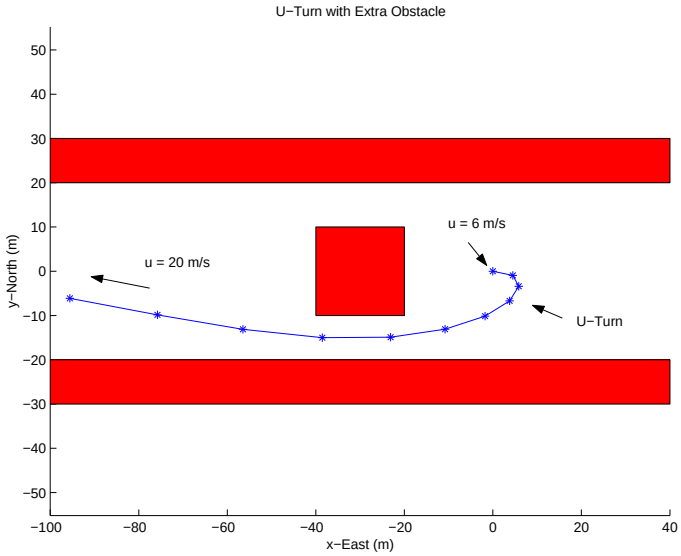


Figure 3-12: Scenario 4: The helicopter is initially in $(0, 0)$ flying at 6 m/s and needs to reverse direction towards $(-100, 0)$ as quickly as possible. To avoid the obstacle in the middle, it decides to make wider a U-turn.

In the last two examples, shown in Figures 3-13 and 3-14, we narrowed the width of the corridor to 20 m and 10 m respectively. The helicopter starts at 12 m/s in both cases. In the 20 m scenario, the vehicle first turns northeast such that it can still safely execute the split-S. In the 10 m scenario, however, because of the 5 m safety boundaries along the walls, the only feasible option is to slow down and make a turn on the spot.

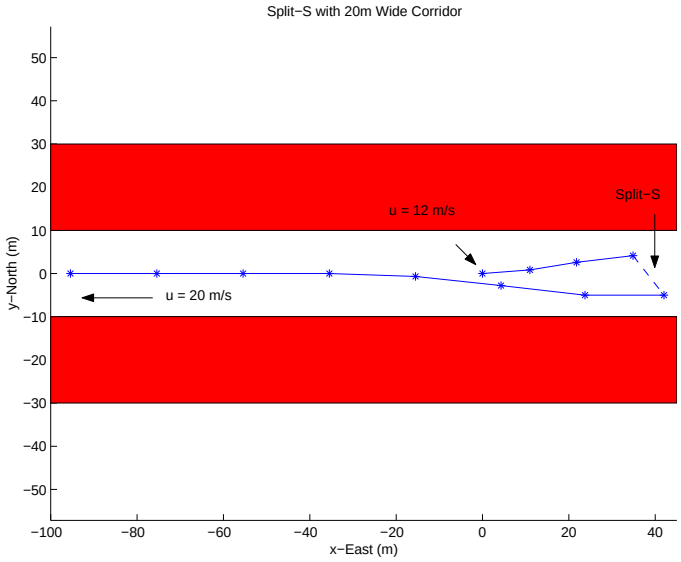


Figure 3-13: Scenario 5: The helicopter is initially in $(0, 0)$ flying at 12 m/s and needs to reverse direction towards $(-100, 0)$ as quickly as possible. Because of the narrower corridor, the vehicle first turns northeast before executing the split-S.

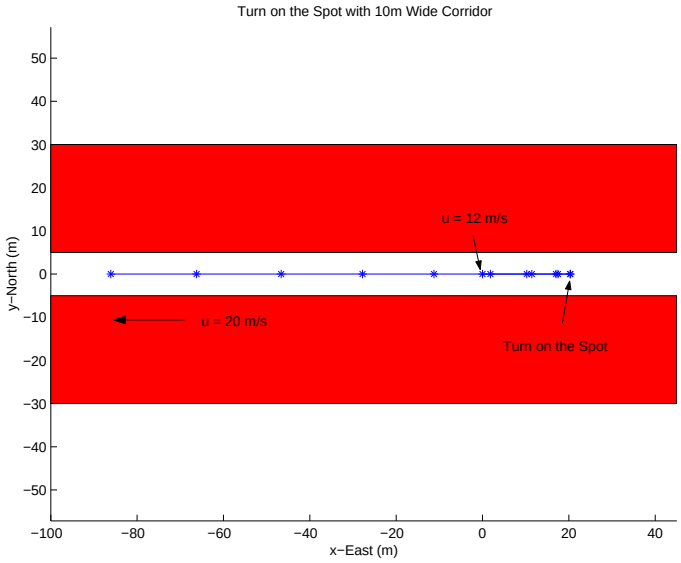


Figure 3-14: Scenario 6: The helicopter is initially in $(0, 0)$ flying at 12 m/s and needs to reverse direction towards $(-100, 0)$ as quickly as possible. The only feasible option is to slow down and make a turn on the spot.

Notice that in the above examples, the vehicle was not constrained to hit the waypoint exactly. If desired, however, this could be achieved by switching to a different cost function and/or by including additional arrival constraints as soon as the waypoint is reachable within the planning horizon. The option of changing cost functions between two iterations is discussed in more detail in Section 3.6 about different planning modes.

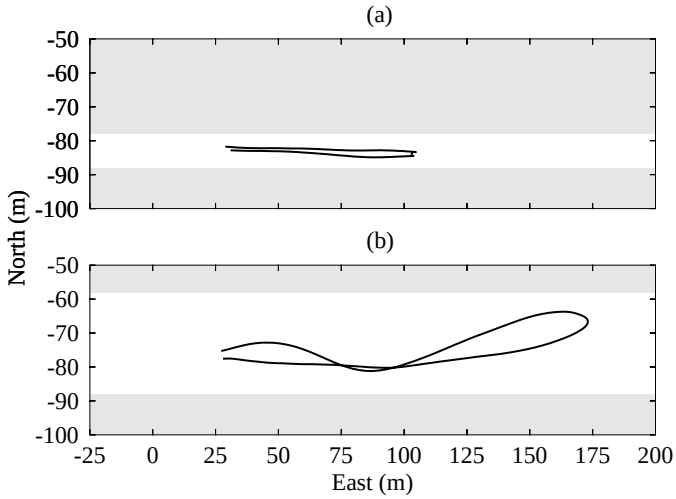


Figure 3-15: Hardware in the loop experiments. The helicopter is flying between two obstacles, initially heading east at 10 m/s. Figure (a) shows the helicopter reversing direction by slowing down till hover and turning on the spot. In Figure (b), it performs a hammerhead. The different actions depend on the width of the corridor.

3.5.3 Hardware in the Loop Experiments

The optimal turn-around scenario was also implemented in the X-Cell's hardware-in-the-loop simulator. For the experiments we considered one LTI-mode up to 18 m/s and the hammerhead as the only agile maneuvering option. The reason for this restricted library was the limited computation power of the onboard computer and the use of the open source MILP solver GLPK [86], which is less powerful than CPLEX. The GLPK solver was integrated with the helicopter guidance and flight control software in the real-time operating system QNX 4.25. It ran on a DSP Design TP400B single board computer with a National Semiconductor Geode GX1 chip, operating at 300 MHz. The MILP solver was implemented as an independent low priority process, which interacted with the remaining flight code through shared memory. This architecture ensured that the primary flight control system was not affected by the computationally demanding solver.

Figure 3-15 shows two real-time trajectories resulting from when the helicopter is initially flying east at 10 m/s. In the first scenario, the lateral displacement was limited to 10 m, while in the second, it was allowed to be 30 m. The plots start at the time at which the threat is detected, and end when the helicopter reaches 14 m/s in the opposite direction. In the first case, the helicopter opted for a turn on the spot, while in the second scenario, the fastest action was to accelerate and perform the hammerhead.

3.6 Practical Considerations

When planning in a receding horizon fashion, it may be possible to take some decisions outside of the actual trajectory optimization problem. Decoupling specific elements of the planning problem by introducing a decision hierarchy can save significant computation time. This can be performed by tailoring the formulation of the trajectory optimization to the requirements set by a specific context. In some situations, it may indeed be possible to

recognize *a priori* certain features of the planning problem based on the vehicle's operating condition and aspects of the environment. For example, in scenario 6 from above, the split-S could be ruled out as a feasible option and removed from the problem formulation ahead of time. More generally, if it is possible to determine whether the vehicle state and environment preclude certain types of maneuvers or LTI conditions, a planning mode with a pruned set of maneuvers and LTI modes can be used, thus reducing the problem complexity and required computation time. Furthermore, an appropriate discretization level can be used, depending on whether the context requires less or more precise trajectories.

These considerations are of importance in the implementation of a real-time guidance system. Exploiting a priori knowledge can also be used in the computation of a heuristic cost-to-go function. We believe that the most effective implementation will be one in which an online receding horizon approach is combined with an offline cost-to-go calculation. Several (parameterized) cost-to-go functions could be pre-computed for different, often recurring situations, such as turning around a street corner in an urban environment. These cost-to-go functions can then be incorporated in the online optimization problem as terminal costs at the end of the planning horizon.

Such different planning modes would also allow the trajectory generation software to switch between inertial and body-fixed frame models of the vehicle. An inertial model, for instance, could be used to compute "high resolution" waypoints in a cluttered environment, which are then followed by solving MILP problems in a body-fixed reference frame. The MILP optimization itself could then act as the waypoint controller.

3.7 Conclusion

This chapter presented a hybrid control architecture and model for autonomous trajectory planning of agile vehicles by combining multiple velocity control modes with a maneuver scheduler. The former provide the flexibility to precisely navigate between waypoints in a cluttered environment, while the latter enables execution of pre-programmed maneuvers at the limit of the vehicle capabilities. The closed-loop dynamics under this control architecture were described by a simple hybrid model consisting of a set of LTI modes and discrete, fixed-duration state transitions. Using this description of the dynamics, optimal trajectory planning through a cluttered environment was formulated as a mixed-integer linear program. The framework was worked out in detail for the case of a small-scale helicopter model based on MIT's aerobatic X-Cell. Results for several receding horizon scenarios were presented that illustrate the real-time applicability of the approach. In addition, some practical considerations regarding an efficient implementation were discussed.

Chapter 4

Trajectory Planning with Feasibility and Safety Guarantees

This chapter extends the basic receding horizon trajectory problem presented in Chapter 2 to account for feasibility and safety guarantees. We consider the case of a single vehicle navigating through a cluttered environment which is only known within a certain detection radius around the vehicle. A receding horizon strategy is presented with hard terminal constraints that guarantee feasibility of the trajectory planning problem at all future time steps. The trajectory computed at each iteration is constrained to end in a feasible invariant set, in which the vehicle can remain for an indefinite period of time. These invariant sets need not be known ahead of time and are implicitly computed online. The principle is applied to the case of a UAV with limited turn rate and minimum speed requirements, for which feasible invariant sets are derived in the form of loiter circles. Safety is guaranteed by further constraining the terminal feasible invariant set to contain an entry state to a backtrack pattern that brings the vehicle back to where it started. Example scenarios are presented that illustrate the necessity of the feasibility and safety constraints when the knowledge of the environment is limited and/or hard real-time restrictions are given.

4.1 Introduction

In scenarios where the environment is not fully characterized before the mission and explored online, the basic trajectory planning problem presented in Chapter 2 may not always have a feasible solution. Indeed, despite the hard obstacle and collision avoidance constraints, the basic receding horizon strategies (2.3)-(2.9) and (2.10)-(2.15) for multiple and single vehicles respectively have no safety guarantees regarding avoidance of obstacles and collisions in the future. Namely, the algorithm may fail to provide a solution in future time steps due to obstacles and other vehicles that are located beyond the planning radius of the vehicle. For example, the trajectory planned at the current iteration may approach an unknown obstacle that is located just outside its detection radius or planning horizon too closely, and may not have sufficient braking power or turn capacity to compensate for it over the next time steps. This translates into the optimization problem becoming infeasible at a certain time step in the future, indicating that the vehicle is on a collision course.

This chapter therefore develops additional constraints that guarantee *feasibility* and *safety* of the receding horizon planning problem at all future iterations. The constraints are expressed as affine transformations of the last state in the planning horizon, and, as such, are

implicit to the optimization problem. They imply the existence of an invariant set [64] that is parameterized in terms of the terminal state of the trajectory at each receding horizon iteration but is not pre-computed. In this respect, our method to maintaining feasibility and safety is fundamentally different from the existing approaches in the literature [120, 146, 102]. These methods are based on constraining the online trajectory to end in a reachable invariant set that is computed offline [83, 97], for example by solving a Hamilton-Jacobi equation [4, 96]. As such, the environment must either be known ahead of time or the sets must be conservative to handle the uncertainty in the environment. Our method does not require any off-line computation, is more flexible in its formulation and allows for less conservative trajectories. Furthermore, at each time step it will provide an *a priori* known trajectory that can serve as a safe backup plan in case a new solution cannot be found in time. This "rescue" trajectory will be updated at each receding horizon iteration.

The type of feasibility that will be ensured by our formulation is feasibility with respect to geometric constraints that are imposed by the environment. This is different from the robust model predictive control literature which mainly studies the problem of maintaining feasibility against model uncertainties or external disturbances acting on the vehicle. A survey of such methods can be found in [13]. Among the existing methods, we mention min-max approaches [134] and the use of linear matrix inequalities [66]. Although such methods could be used to guarantee additional robust feasibility against disturbances, throughout this chapter, we only consider the nominal planning problem. Robust feasibility of the basic trajectory planning using constraint tightening [46] is covered in [112], whereas robustness of the safe planning formulation developed in this chapter is still a topic of ongoing research [71, 70].

Furthermore, unlike other constrained model predictive control formulations and problems [29, 136, 135, 91], feasibility of the receding horizon trajectory planning problem through a partially unknown environment does not automatically imply stability in the sense of reaching the goal. Unknown obstacles in the environment may indeed prevent the vehicle from reaching its desired location. Stability is therefore an issue related to the environment modeling and cost-to-go formulation [7] rather than a constraint related problem. As such, our primary concern in this chapter is to guarantee *safety* of a mission rather than *completion* of the mission, which may be infeasible.

The outline of the chapter is as follows. Section 4.2 first presents some scenarios that highlight the need for safety constraints in the basic receding horizon formulation. Section 4.3 then defines the concept of a terminal feasible invariant set and introduces a receding formulation with feasibility guarantees. Next, an example in the form of loiter circles is given in Section 4.4, along with simulation results. Section 4.5 then defines safety and introduces additional constraints to obtain a safe receding horizon planning strategies.

4.2 Unsafe Scenarios

Consider Example 1 from Section 2.4 again. Assume that the vehicle has a detection radius of 30 m and that obstacles beyond that radius are unknown. As depicted in Figure 4-1, consider now the case where the corridor through which the aircraft is flying is obstructed. Then, because of its 30 m detection radius and corresponding limited knowledge of the obstacle field, the MILP optimization guides the vehicle into the concavity from which it cannot exit. This is due to its limitation on turn rate and minimum speed, resulting in a minimum turn radius that is larger than the available maneuver space. This observation

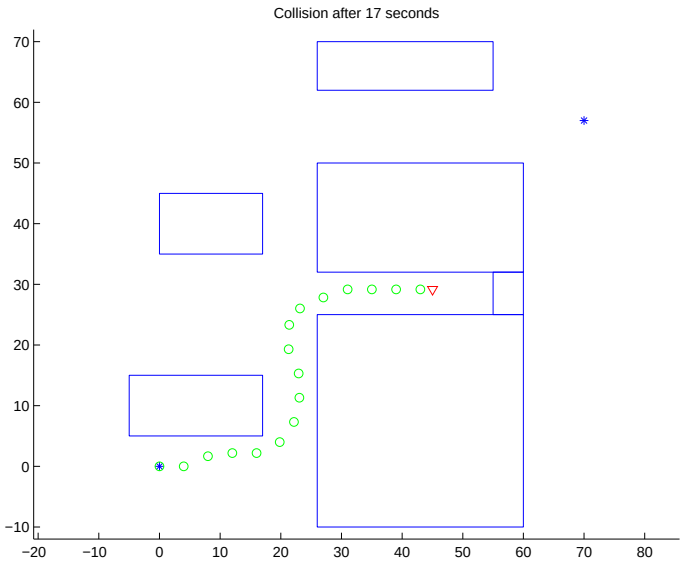


Figure 4-1: The aircraft is initially in the origin, flying east at 4 m/s, and has to maneuver to position $(70, 57)$ m. After 17 s, the MILP becomes infeasible, corresponding to the aircraft colliding with the obstacles.

translates into the MILP becoming infeasible after 17 s and a crash of the aircraft against an obstacle.

In the multi-vehicle scenario from Section 2.4 (Example 2), decreasing the number of time steps accounted for in the cost function while keeping the number of constraints the same also results in an infeasibility of the collision avoidance constraints. The cost function will drive the vehicles too close to each others until they can no longer turn away fast enough. This is shown in Figure 4-2: just before halfway towards the destination points, the problem becomes infeasible and an intersection of the avoidance zones can no longer be avoided. The multi-vehicle safety problem will be tackled in Chapter 5; in this chapter, we will first discuss the single vehicle case.

4.3 Feasibility Constraints

4.3.1 Feasible Invariant Sets

The preceding examples indicate that when a vehicle is maneuvering through an environment that is only partially known, additional constraints are needed that guarantee its safety at future time steps. One could claim that by choosing a more sophisticated cost-to-go function that assigns a higher cost to unsafe regions in the state space, situations as the ones described above could be avoided automatically. However, our assumption is that no or only partial information about the environment is known beyond the detection region of the vehicle. As such, a cost-to-go construction based on knowledge of all obstacles as in [7] can only be applied locally, without guarantees for the future.

To precisely capture what we understand by safety and how it is related to the feasibility of the trajectory planning problem, we introduce some definitions:

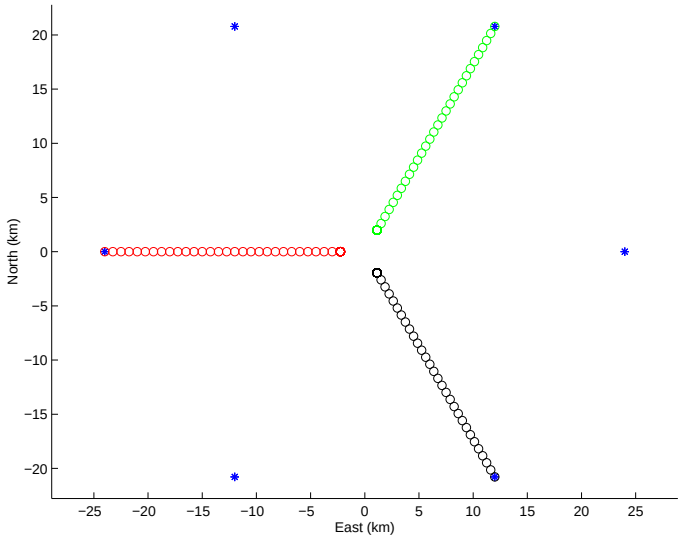


Figure 4-2: The aircraft are initially positioned along a circle and need to fly to the opposite side. Near the origin, the MILP becomes infeasible: the aircraft have approached each other too closely and an intersection of their avoidance zones cannot be avoided.

Definition 4.1 (Feasible Invariant Set): *We define a* feasible invariant set $\mathcal{S}(t)$ *as a state or periodic sequence of* $I_p < \infty$ *states starting at time* t *that is dynamically feasible, obstacle-free, and in which the vehicle can remain for an indefinite period of time:* $\mathcal{S}(t) = \{\boldsymbol{x}(t+s|t) \in \mathbb{R}^n, s = 0, \ldots, I_p \mid \boldsymbol{x}(t+s|t) \in \mathcal{X}(t), \boldsymbol{x}(t+s|t) \notin \mathcal{O}(t), \boldsymbol{x}(t+I_p|t) = \boldsymbol{x}(t|t), \forall s < I_p \, \exists \boldsymbol{u}(t+s|t) \in \mathcal{U}(t-1) : \boldsymbol{x}(t+s+1|t) = \boldsymbol{A}\boldsymbol{x}(t+s|t) + \boldsymbol{B}\boldsymbol{u}(t+s|t)\}$.

Examples of such feasible invariant sets are stop states for a ground rover, hover for a helicopter, and loiter patterns for fixed-wing UAVs that lie in a region of the environment that is fully characterized and will remain free of obstacles in the future. Regardless of how such a set is determined, it is clear that once the vehicle enters the safe invariant set, there will always exist a dynamically feasible, obstacle-free trajectory at all future time steps. Therefore, if at each receding horizon iteration, the trajectory of the vehicle is constrained to terminate in such a set, (nominal) feasibility at the next time step will be guaranteed. We call this set a *terminal* feasible invariant set:

Definition 4.2 (Terminal Feasible Invariant Set): *We define a* terminal feasible invariant set $\mathcal{S}(t+T|t)$ *as a feasible invariant set starting at the last state* $\boldsymbol{x}(t+T|t)$ *of the receding horizon trajectory planning problem at time* t*:* $\mathcal{S}(t+T|t) = \{\boldsymbol{x}_{\mathcal{S}(t)}(t+T+s|t) \in \mathbb{R}^n, s = 0, \ldots, I_p \mid \boldsymbol{x}_{\mathcal{S}(t)}(t+T|t) \equiv \boldsymbol{x}(t+T|t), \boldsymbol{x}_{\mathcal{S}(t)}(t+T+s|t) \in \mathcal{X}(T), \boldsymbol{x}_{\mathcal{S}(t)}(t+T+s|t) \notin \mathcal{O}(t), \boldsymbol{x}_{\mathcal{S}(t)}(t+T+I_p|t) = \boldsymbol{x}_{\mathcal{S}(t)}(t+T|t), \forall s < I_p \, \exists \boldsymbol{u}_{\mathcal{S}(t)}(t+T+s|t) \in \mathcal{U}(T-1) : \boldsymbol{x}_{\mathcal{S}(t)}(t+T+s+1|t) = \boldsymbol{A}\boldsymbol{x}_{\mathcal{S}(t)}(t+T+s|t) + \boldsymbol{B}\boldsymbol{u}_{\mathcal{S}(t)}(t+T+s|t)\}$.

Notice that the kino-dynamic constraint sets $\mathcal{X}(T)$ and $\mathcal{U}(T-1)$ for the terminal feasible invariant set are now the ones corresponding to the last state and input of the planning horizon. The obstacle set $\mathcal{O}(t)$ is still the one as characterized at time t, because that is when the trajectory terminating in $\mathcal{S}(t+T|t)$ is planned.

4.3.2 Feasible Receding Horizon Planning Problem

Accounting for the terminal feasible invariant set in the receding horizon trajectory optimization yields the following extended formulation of the basic planning problem (2.10)-(2.15):

$$J_T^* = \min \sum_{k=0}^{T-1} \ell_k(\boldsymbol{x}(t+k|t), \boldsymbol{u}(t+k|t), \boldsymbol{x}_f) + f_T(\boldsymbol{x}(t+T|t), \boldsymbol{x}_f) \tag{4.1}$$

subject to:

$$\begin{aligned}
\boldsymbol{x}(t+k+1|t) &= \boldsymbol{A}\boldsymbol{x}(t+k|t) + \boldsymbol{B}\boldsymbol{u}(t+k|t),\ k = 0, \ldots, T-1 && (4.2)\\
\boldsymbol{x}(t|t) &= \hat{\boldsymbol{x}}(t|t-1) && (4.3)\\
\boldsymbol{x}(t+k|t) &\in \mathcal{X}(k) \supseteq \mathcal{X}(k+1),\ k = 1, \ldots, T && (4.4)\\
\boldsymbol{u}(t+k|t) &\in \mathcal{U}(k) \supseteq \mathcal{U}(k+1),\ k = 0, \ldots, T-1 && (4.5)\\
\boldsymbol{p}(t+k|t) &\notin \mathcal{O}_a(t),\ k = 1, \ldots, T && (4.6)\\
\boldsymbol{x}(t+T|t) &\in \mathcal{S}(t+T|t) && (4.7)
\end{aligned}$$

It will be referred to as $\mathrm{P}_F(\boldsymbol{x}(t), \mathcal{X}(k), \mathcal{U}(k), \mathcal{O}(t), \mathcal{S}(t+T|t))$. Notice the additional superset structure $\mathcal{X}(k) \supseteq \mathcal{X}(k+1)$ and $\mathcal{U}(k) \supseteq \mathcal{U}(k+1)$ for the state and input constraint sets. Together with the terminal feasible invariant set, these additional requirements are needed to maintain a priori feasibility of the optimization problem $\mathrm{P}_F(\boldsymbol{x}(t+1), \mathcal{X}(k), \mathcal{U}(k), \mathcal{O}(t+1), \mathcal{S}(t+1+T|t+1))$ at the next receding horizon iteration. The feasibility property is formalized in the following theorem:

Theorem 4.1: *Feasibility of the trajectory optimization problem* $\mathrm{P}_F(\boldsymbol{x}(t), \mathcal{X}(k), \mathcal{U}(k), \mathcal{O}(t), \mathcal{S}(t+T|t))$ *at time* t *implies feasibility of the problem* $\mathrm{P}_F(\boldsymbol{x}(t+1), \mathcal{X}(k), \mathcal{U}(k), \mathcal{O}(t+1), \mathcal{S}(t+1+T|t+1))$ *at time* $t+1$.

Proof: The proof consists of explicitly constructing a feasible solution to the next optimization problem $\mathrm{P}_F(\boldsymbol{x}(t+1), \mathcal{X}(k), \mathcal{U}(k), \mathcal{O}(t+1), \mathcal{S}(t+1+T|t+1))$. Assume that a feasible solution to $\mathrm{P}_F(\boldsymbol{x}(t), \mathcal{X}(k), \mathcal{U}(k), \mathcal{O}(t), \mathcal{S}(t+T|t))$ at time t is given by the input sequence $\boldsymbol{u}^*(t|t), \boldsymbol{u}^*(t+1|t), \ldots, \boldsymbol{u}^*(t+T-1|t)$ and corresponding state sequence $\boldsymbol{x}^*(t|t), \boldsymbol{x}^*(t+1|t), \ldots, \boldsymbol{x}^*(t+T-1|t), \boldsymbol{x}^*(t+T|t)$, where $\boldsymbol{x}^*(t+T|t) \in \mathcal{S}(t+T|t)$. Per definition of $\mathcal{S}(t+T|t)$, there exists an input $\boldsymbol{u}^*(t+T) \in \mathcal{U}(T-1)$ such that $\boldsymbol{x}^*(t+T+1) = \boldsymbol{A}\boldsymbol{x}(t+T|t) + \boldsymbol{B}\boldsymbol{u}^*(t+T)$ with $\boldsymbol{x}^*(t+T+1) \in \mathcal{X}(T)$, $\boldsymbol{x}^*(t+T+1) \in \mathcal{S}(t+T|t)$ and $\boldsymbol{x}^*(t+T+1) \notin \mathcal{O}(t)$. Let a candidate solution to problem $\mathrm{P}_F(\cdot)$ at time $t+1$ then be given by $\boldsymbol{u}^*(t+1|t+1) = \boldsymbol{u}^*(t+1|t) \in \mathcal{U}(1) \subset \mathcal{U}(0), \boldsymbol{u}^*(t+2|t+1) = \boldsymbol{u}^*(t+2|t) \in \mathcal{U}(2) \subset \mathcal{U}(1), \ldots, \boldsymbol{u}^*(t+T-1|t+1) = \boldsymbol{u}^*(t+T-1|t) \in \mathcal{U}(T-1) \subset \mathcal{U}(T-2), \boldsymbol{u}^*(t+T|t+1) = \boldsymbol{u}^*(t+T) \in \mathcal{U}(T-1)$ and $\boldsymbol{x}^*(t+1|t+1) = \boldsymbol{x}^*(t+1|t), \boldsymbol{x}^*(t+2|t+1) = \boldsymbol{x}^*(t+2|t) \in \mathcal{X}(2) \subset \mathcal{X}(1), \ldots, \boldsymbol{x}^*(t+T|t+1) = \boldsymbol{x}^*(t+T|t) \in \mathcal{X}(T) \subset \mathcal{X}(T-1), \boldsymbol{x}^*(t+1+T|t+1) = \boldsymbol{x}^*(t+T+1) \in \mathcal{X}(T)$. The state space equations and state and input constraints are thus satisfied. Furthermore, by letting $\mathcal{S}(t+1+T|t+1) = \mathcal{S}(t+T|t)$, we also have $\boldsymbol{x}^*(t+1+T|t+1) \in \mathcal{S}(t+1+T|t+1)$. Finally, because we are assuming a static environment and the new trajectory does not reach further in distance than the previous one (since it terminates in the same feasible invariant set), no new obstacles will have to be accounted for and the avoidance constraints are still satisfied. The candidate solution is thus feasible which proofs the feasibility of $\mathrm{P}_F(\boldsymbol{x}(t+1), \mathcal{X}(k), \mathcal{U}(k), \mathcal{O}(t+1), \mathcal{S}(t+1+T|t+1))$. □

Given a feasible trajectory at the current time step t, an *a priori* feasible solution to the next optimization problem at $t+1$ is thus given by the remaining part of the current trajectory with an additional state in the current terminal feasible invariant set $\mathcal{S}(t+T|t)$. This solution can act as a warm start at the next receding horizon iteration and serve as a feasible backup plan in case no better trajectory can be found within the real-time limits of the guidance system. Next, we make the following observation:

Corollary 4.1: *By recursion, feasibility of the trajectory optimization problem* $\mathrm{P}_F(\cdot)$ *at* $t=0$ *implies feasibility of* $\mathrm{P}_F(\cdot)$ *at all future time steps.*

Maintaining feasibility over all time steps is an important property for receding horizon planning strategies. Besides allowing for warm starts in the optimization, it is fundamental in the study of robust planning techniques and the correctness proofs thereof [71, 70].

4.4 Feasible Invariant Sets as Affine Transformations

4.4.1 Terminal Feasible Invariant Set Modeling

So far, we have not elaborated on how the terminal feasible invariant sets for problem (4.1)-(4.7) are determined. In general, they will be expressed as affine transformations on the last state in the planning horizon. Binary variables are then used again to ensure the associated obstacle avoidance requirement. In this section, we specialize the approach to the case of an aircraft that is constrained by the kino-dynamic inequalities (2.21)-(2.23). Because of the minimum speed constraint, a natural feasible invariant set is a circular loiter pattern. If the trajectory computed at each time step ends in either a left or right turning loiter circle that does not intersect any of the obstacles, the optimization problem is guaranteed to be feasible at each future time step. This implies that the aircraft will be able to physically avoid all obstacles within its detection radius, and if necessary, be able to transition to and remain on its obstacle-free loiter circle.

Assuming that the aircraft behaves like a double integrator, the radius R of the smallest possible loiter circle at a given speed v corresponds to $R = cv^2$. Here c is an aircraft specific parameter associated with the maximum available lateral acceleration or force. As depicted in Figure 4-3, to describe the loiter circle as a function of the last state $\boldsymbol{x}(T) = [x(T)\, y(t+T)\, \dot{x}(T)\, \dot{y}(T)]'$ in the planning horizon, we need to find the vectors $(\boldsymbol{p}(T) - \boldsymbol{p}_R)$ and $(\boldsymbol{p}(T) - \boldsymbol{p}_L)$. Here, $\boldsymbol{p}(T) = [x(T)\ y(T)]'$ denotes the ingress position of the loiter, and $\boldsymbol{p}_R$ and $\boldsymbol{p}_L$ are the center points of the right and left circle respectively. The latter are scaled versions $\alpha \boldsymbol{v}^{\perp}(T)$ and $-\alpha \boldsymbol{v}(T)^{\perp}$ of the orthogonal complement $\boldsymbol{v}^{\perp}(T) = [-\dot{y}(T)\ \dot{x}(T)]'$ of $\boldsymbol{v}(T) = [\dot{x}(T)\ \dot{y}(T)]'$.

The scaling factor α has a lower and upper bound, corresponding to the minimum and maximum allowed ingress velocity. With $||\boldsymbol{v}(T)||$ the magnitude of $\boldsymbol{v}(T)$, we have $\alpha = \frac{R}{||\boldsymbol{v}(T)||} = c||\boldsymbol{v}(T)||$ and thus $\alpha_{min} = cv_{min} \leq \alpha \leq \alpha_{\max} = cv_{max}$. However, to avoid quadratic constraints on $\boldsymbol{v}(T)$, we use a constant scaling factor α_c. By conservatively setting $\alpha_c = \alpha_{max}$, the radius of the loiter circles will be larger than necessary for ingress velocities lower than v_{max}, which corresponds to not applying the maximum available lateral thrust. Since this is an over-approximation of a safety condition, however, we are only giving up some performance rather than safety.

By introducing a rotation matrix $\mathbf{R}(\theta)$, any point $\boldsymbol{p}_R(\theta)$ along the right loiter circle

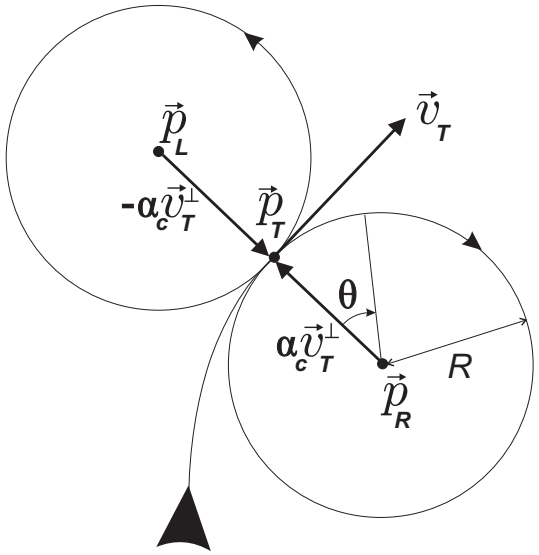


Figure 4-3: Safe trajectory ending in either a right or left turning loiter circle.

$\mathcal{C}_R(\boldsymbol{x}(T))$ can then be expressed as:

$$\begin{aligned} \boldsymbol{p}_R(\theta) &= \boldsymbol{p}_R + \mathbf{R}(\theta)(\alpha_c \boldsymbol{v}^{\perp}(T)) \\ &= (\boldsymbol{p}(T) - \alpha_c \boldsymbol{v}^{\perp}(T)) + \mathbf{R}(\theta)(\alpha_c \boldsymbol{v}^{\perp}(T)) \\ &= \boldsymbol{p}(T) + \alpha_c(\boldsymbol{R}(\theta) - \boldsymbol{I})\boldsymbol{v}^{\perp}(T) \\ &= \begin{bmatrix} x(T) \\ y(T) \end{bmatrix} + \alpha_c \begin{bmatrix} \cos\theta - 1 & -\sin\theta \\ \sin\theta & \cos\theta - 1 \end{bmatrix} \begin{bmatrix} -\dot{y}(T) \\ \dot{x}(T) \end{bmatrix} \end{aligned} \tag{4.8}$$

Similarly, any point $\boldsymbol{p}_{L,\theta}$ along the left loiter circle $\mathcal{C}_L(\boldsymbol{x}(T))$ is given by:

$$\begin{aligned} \boldsymbol{p}_L(\theta) &= \boldsymbol{p}_L - \boldsymbol{R}(\theta)(\alpha_c \boldsymbol{v}^{\perp}(T)) \\ &= (\boldsymbol{p}(T) + \alpha_c \boldsymbol{v}^{\perp}(T)) - \boldsymbol{R}(\theta)(\alpha_c \boldsymbol{v}^{\perp}(T)) \\ &= \boldsymbol{p}(T) - \alpha_c(\boldsymbol{R}(\theta) - \boldsymbol{I})\boldsymbol{v}^{\perp}(T) \\ &= \begin{bmatrix} x(T) \\ y(T) \end{bmatrix} - \alpha_c \begin{bmatrix} \cos\theta - 1 & -\sin\theta \\ \sin\theta & \cos\theta - 1 \end{bmatrix} \begin{bmatrix} -\dot{y}(T) \\ \dot{x}(T) \end{bmatrix} \end{aligned} \tag{4.9}$$

4.4.2 Sampling Points Requirements

Maintaining feasibility now comes down to ensuring that either the left or right loiter circle does not overlap with any of the obstacles that are located within the detection radius of the vehicle. This can be achieved by sampling both circles for fixed values of θ and by introducing avoidance constraints similar to (2.29). To strictly follow the approach used to guarantee feasibility in the proof of Theorem 4.1, the sampling points should correspond to actual time steps Δt, yielding $N_{\Delta t} = \lceil 2\pi/(\omega_{max}\Delta t) \rceil$ points and a sample angle of $\theta_s = \frac{2\pi}{N_{\Delta t}}$. Taking the ceiling in the expression for $N_{\Delta t}$ is required to guarantee that $\boldsymbol{p}_L(N_{\Delta t}\theta_s) = \boldsymbol{p}(T)$, conformable to the definition of a terminal feasible invariant set. However, we could also sample the loiter patterns geometrically, i.e., with a sampling angle that does not correspond to an actual time step along the circles. As long as the full circle is obstacle-free, this does not jeopardize the feasibility of the planning problem at the next time step. Alternatively speaking, the sampling comes down to a rescaling of the time step once the vehicle is in

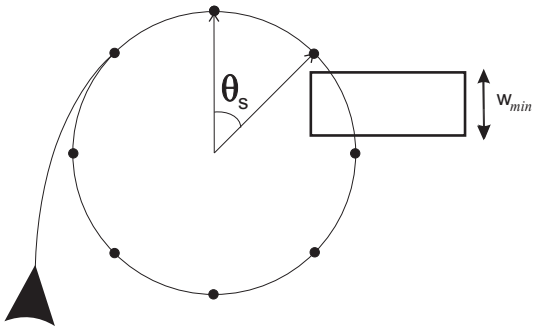


Figure 4-4: Situation where undersampling of the loiter circle leads to a safety violation. Although the obstacle avoidance constraints for the sample points are satisfied, the circle intersects the obstacle.

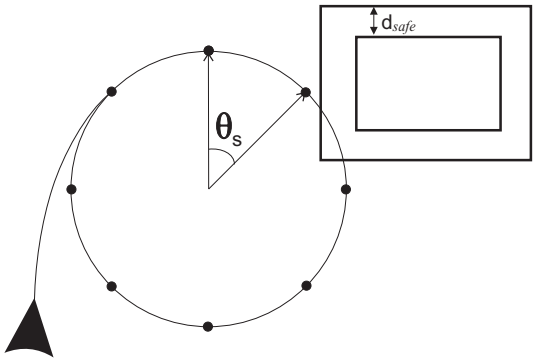


Figure 4-5: Situation where the loiter circle cuts the corner of an obstacle. This situation can be avoided by enlarging the obstacles with a safety boundary d_{safe}.

the feasible invariant set. It allows us to plan larger invariant sets with a smaller number of variables.

As before, ensuring obstacle avoidance for sample points along the circle does not guarantee that the loiter circle does not intersect obstacles in the segments between the sample points. Consider for example the situation depicted in Figure 4-4: although the avoidance constraints for all sample points are satisfied, the circle cuts through the obstacle because the sample angle spacing is too coarse. This type of under-sampling can be avoided by choosing a minimum number of sampling points N as follows:

$$N \geq N_{\min} = \frac{\pi}{\arcsin(\frac{w_{\min}}{2r_{\max}})} \tag{4.10}$$

Here $w_{\min}$ denotes the width of the narrowest obstacle, and $r_{\max} = cv_{\max}^2$ is the radius of the largest loiter circle. The derivation of this condition is based on the insight that the maximum spacing in distance between the sample points along the largest circle should not exceed $w_{\min}$. As such, (4.10) is only necessary when $w_{\min} \leq 2r_{\max}$.

If $w_{\min} > 2r_{\max}$, a situation like the one in Figure 4-4 cannot occur, and therefore no minimum number of sample points is required. However, as illustrated in Figure 4-5, the loiter circles can now cut the corners of obstacles. Nevertheless, avoidance can still be guaranteed by enlarging the obstacles with a safety boundary d_{safe} such that the circle can

enter the boundary, but does not intersect the actual obstacle. Using basic geometry, one can derive the following expression for d_{safe} as a function of N:

$$d_{\text{safe}}(N) = \frac{\sqrt{2}}{2} r_{\max} \left(1 + \sin\frac{\pi}{N} - \cos\frac{\pi}{N}\right) < \sqrt{2} r_{\max}$$

Remember that the enlargement principle also holds for the regular part of the trajectory: due to the time discretization with step Δt, each obstacle must be enlarged by $d_{\text{safe}} = \frac{v_{\max}\Delta t}{\sqrt{2}}$.

4.4.3 MILP Formulation

Using the preceding sampling approach, the terminal feasible invariant set constraint (4.7) can be specified by the following loiter conditions:

$$\begin{cases} \mathcal{C}_R(\boldsymbol{x}(T)) = \{(x_{Rj}, y_{Rj})\} \quad \notin \quad \mathcal{O}_s, \ j = 1, \ldots, N \\ \qquad\qquad \text{OR} \\ \mathcal{C}_L(\boldsymbol{x}(T)) = \{(x_{Lj}, y_{Lj})\} \quad \notin \quad \mathcal{O}_s, \ j = 1, \ldots, N \end{cases} \tag{4.11}$$

where the index j indicates the sample point on the circle, and $\mathcal{O}_s$ is the set of obstacles enlarged with d_{safe}. By introducing a binary variable d that selects either the right or left circle, and using (4.8)-(4.9) for the coordinates of the sample points, the avoidance constraints (4.11) for rectangular obstacles can be explicitly written as follows:

$\forall l \in [l, \ldots, L], \forall j \in [1, \ldots, N]$:

$$\begin{cases} x(T) - \alpha_c (\cos j\theta_s - 1)\, \dot{y}(T) - \alpha_c (\sin j\theta_s)\, \dot{x}(T) \\ \qquad\qquad \leq \quad x_{\min,l} + M b_{lj1} + M d \\ -x(T) + \alpha_c (\cos j\theta_s - 1)\, \dot{y}(T) + \alpha_c (\sin j\theta_s)\, \dot{x}(T) \\ \qquad\qquad \leq -x_{\max,l} + M b_{lj2} + M d \\ y(T) - \alpha_c (\sin j\theta_s)\, \dot{y}(T) + \alpha_c (\cos j\theta_s - 1)\, \dot{x}(T) \\ \qquad\qquad \leq \quad y_{\min,l} + M b_{lj3} + M d \\ -y(T) + \alpha_c (\sin j\theta_s)\, \dot{y}(T) - \alpha_c (\cos j\theta_s - 1)\, \dot{x}(T) \\ \qquad\qquad \leq -y_{\max,l} + M b_{lj4} + M d \end{cases} \tag{4.12}$$

AND

$$\begin{cases} x(T) + \alpha_c (\cos j\theta_s - 1)\, \dot{y}(T) + \alpha_c (\sin j\theta_s)\, \dot{x}(T) \\ \qquad\qquad \leq \quad x_{\min,l} + M b_{lj1} + M(1-d) \\ -x(T) - \alpha_c (\cos j\theta_s - 1)\, \dot{y}(T) - \alpha_c (\sin j\theta_s)\, \dot{x}(T) \\ \qquad\qquad \leq -x_{\max,l} + M b_{lj2} + M(1-d) \\ y(T) + \alpha_c (\sin j\theta_s)\, \dot{y}(T) - \alpha_c (\cos j\theta_s - 1)\, \dot{x}(T) \\ \qquad\qquad \leq \quad y_{\min,l} + M b_{lj3} + M(1-d) \\ -y(T) - \alpha_c (\sin j\theta_s)\, \dot{y}(T) + \alpha_c (\cos j\theta_s - 1)\, \dot{x}(T) \\ \qquad\qquad \leq -y_{\max,l} + M b_{lj4} + M(1-d) \end{cases} \tag{4.13}$$

$$\begin{cases} \sum_{k=1}^{4} b_{ljk} \quad \leq \quad 3 \\ \quad b_{ljk},\ d \quad \in \quad \{0,1\} \end{cases} \tag{4.14}$$

Here the index l indicates the obstacles, and $\theta_s = \frac{2\pi}{N}$ is the spacing angle. The obstacle coordinates $(x_{\min}, y_{\min}, x_{\max}, y_{\max})_l$ are those of the enlarged obstacles.

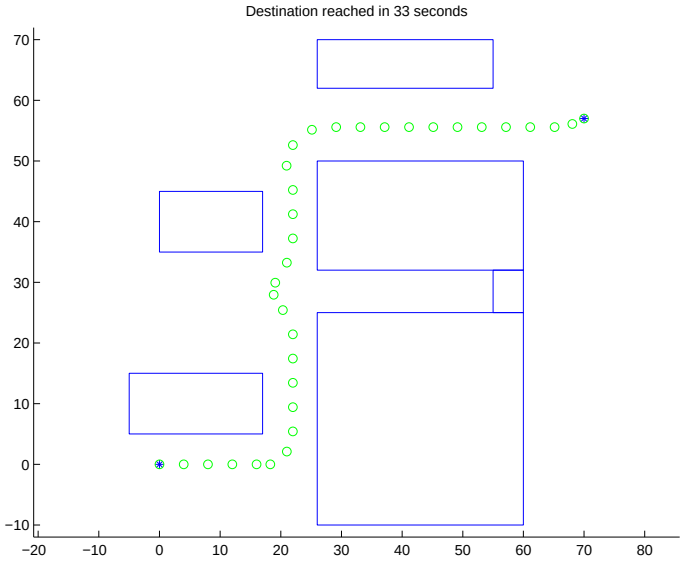


Figure 4-6: Receding horizon trajectory with loiter constraints. Because of the loiter constraints, the UAV avoids the concavity and chooses an alternative route to reach the goal.

4.4.4 Examples

We now apply the loiter circle constraints to the unsafe example of Section 4.2. The planning horizon again contains $T = 6$ time steps of 1 s each. For the loiter circles, we used $N = 8$ sample points. The result is shown in Figure 4-6. Thanks to the loiter constraints, the UAV does not fly into the concavity, but chooses an alternative route to reach the goal. As a result, the MILP remains feasible at all time steps and a crash in the concavity is prevented. The total trajectory time is now 33 ss. The sequence of partial trajectories computed at each time step is depicted in Figure 4-7.

Assume now that the width of the concavity is such that the UAV can make a 180° turn

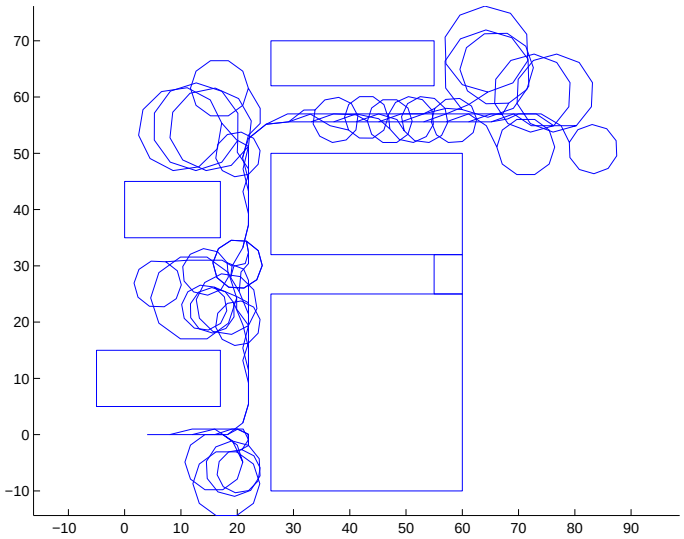


Figure 4-7: Sequence of intermediate receding horizon trajectories ending in loiter circles.

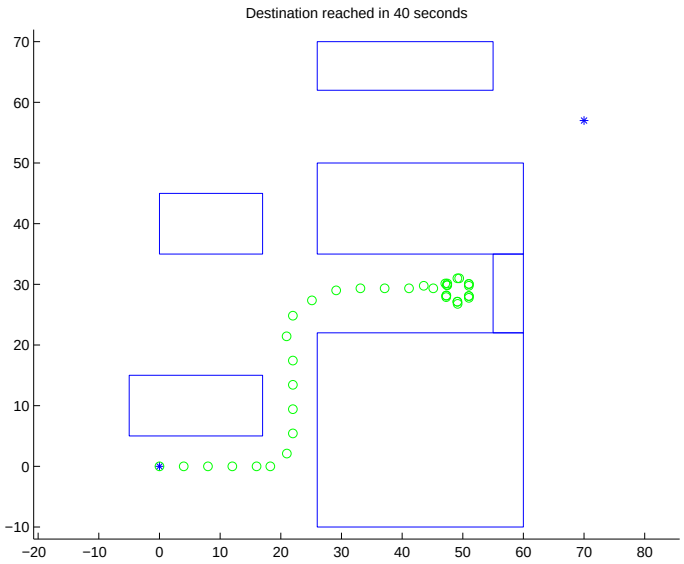


Figure 4-8: Receding horizon trajectory with loiter constraints. Although the UAV enters the concavity, it does not crash and the trajectory ends in a loiter.

in it. In this case, the aircraft does fly into the concavity, but can avoid the obstacle by executing a loiter pattern as displayed in Figures 4-8 and 4-9. Although the mission was not completed, the aircraft remains safe and the optimization problem feasible at all times. As was mentioned earlier, when the vehicle starts loitering, it can use higher level decision logic or an updated cost-to-go function to compute a path out of the concavity.

Finally, Figures 4-10 and 4-11 present a scenario with the same vehicle but with 24 sample points along the loiter circles which are required because of the presence of narrower obstacles in the environment.

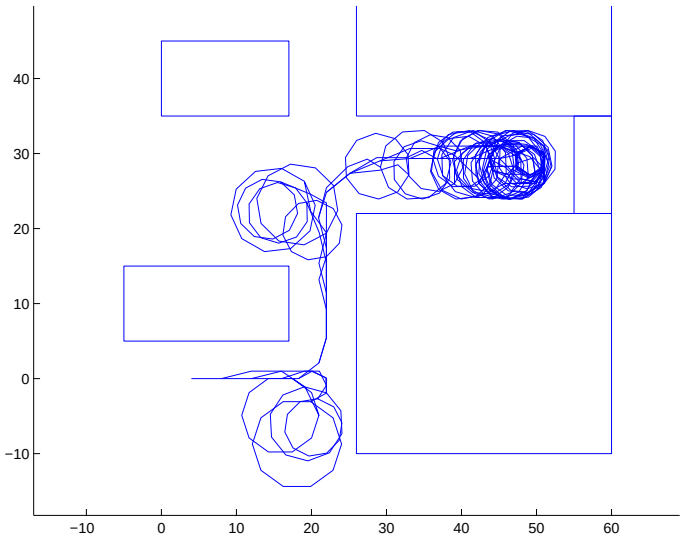


Figure 4-9: Sequence of intermediate receding horizon trajectories for a wider concavity. Although the UAV enters the concavity, it does not crash and the trajectory ends in a loiter.

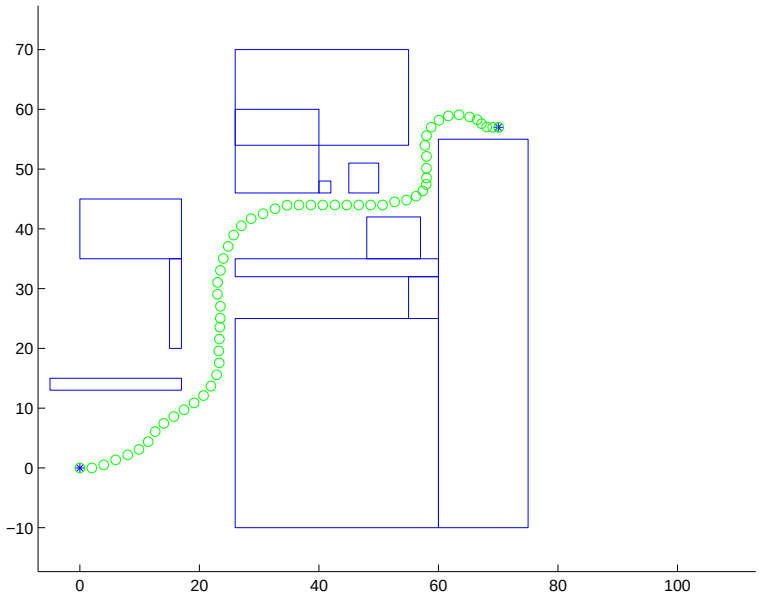


Figure 4-10: Feasible receding horizon trajectory with loiter constraints.

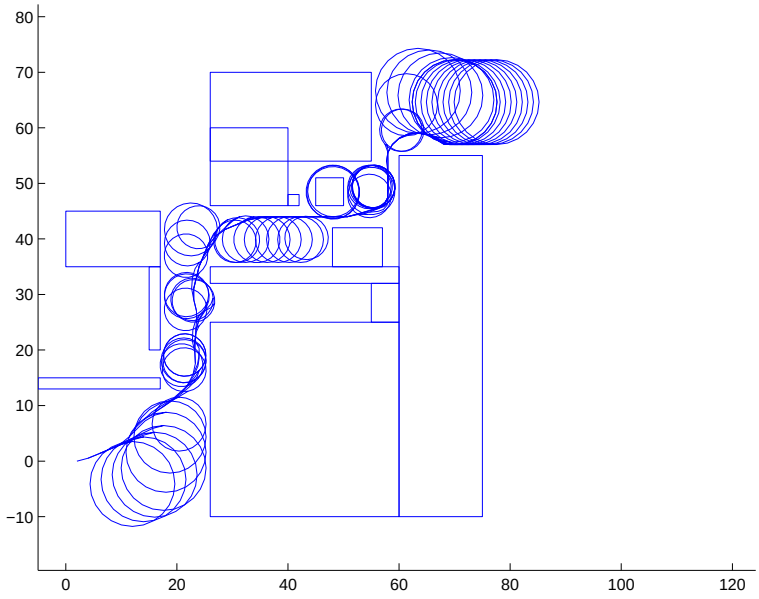


Figure 4-11: Sequence of intermediate receding horizon trajectories with loiter constraints consisting of 24 sample points.

4.5 Safety Constraints

4.5.1 Safety Definitions

Although feasibility of trajectory planning problem $\mathrm{P}_F(\cdot)$ will be guaranteed, actual vehicle safety may not. Indeed, constraining the trajectory to terminate in a feasible invariant set may lead to the vehicle being trapped in that set. For example, a fixed-wing UAV may fly into a concavity in which it can fit a loiter circle without being able to exit from

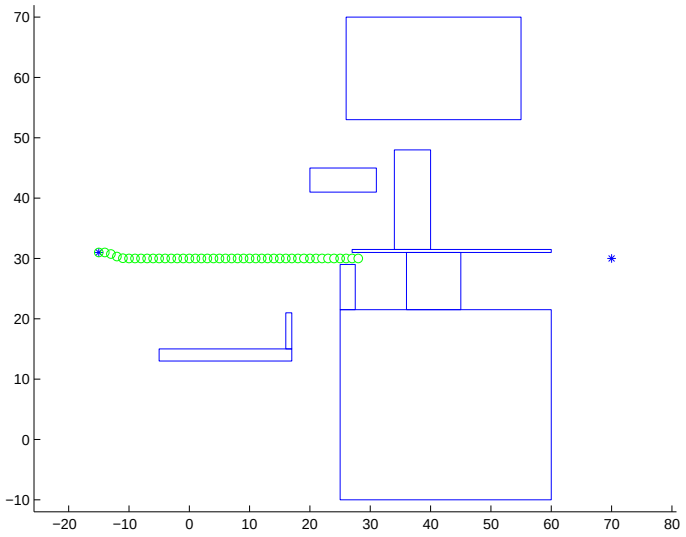


Figure 4-12: Scenario in which the terminal feasible invariant set constraints cannot prevent the UAV from getting trapped inside a concavity. Although feasibility is maintained, actual vehicle safety is not.

that concavity. Such a scenario is shown in Figures 4-12 and 4-13. The parameters of the problem are the following: $v_{max} = v_{min} = 2$ m/s, $\omega_{max} = 30$ deg/s, $T = 7$ time steps with $\Delta t = 0.5$ s and $N_{min} = 24$ loiter sample points. The vehicle starts in (-15 m,31 m) flying east (at 2 m/s) and has to maneuver to (70 m,30 m). Since the cost function is unaware of the shape of the concavity and the loiter circles fit inside, the aircraft is steered into the narrow corridor from which it cannot exit. Although it will not hit any obstacles and feasibility is maintained, it does not have enough space and turn capacity to leave the concavity and is trapped on the circle (until it runs out of fuel).

An absolutely safe planning formulation should therefore account for the possibility to leave the terminal feasible invariant set and return to the position from where the vehicle came. To set up such formulation in a rigorous way, two more concepts have to be introduced:

Definition 4.3 (Return Trajectory): *Define the* return trajectory $\mathcal{T}(t)$ *as the sequence of initial positions of all earlier receding horizons trajectory planning iterations with reversed velocity vectors:* $\mathcal{T}(t) = \{\boldsymbol{x}_{\mathcal{T}(t)}(0) \equiv [\boldsymbol{p}'(t|t) \ -\boldsymbol{v}'(t|t)]', \boldsymbol{x}_{\mathcal{T}(t)}(1) \equiv [\boldsymbol{p}'(t-1|t-1) \ -\boldsymbol{v}'(t-1|t-1)]', \ldots, \boldsymbol{x}_{\mathcal{T}(t)}(t) \equiv [\boldsymbol{p}'(0|0) \ -\boldsymbol{v}'(0|0)]' \in \mathcal{S}(0)\}$. *The return trajectory is stored in memory and gets updated at each receding horizon iteration.*

If the environment is static and provided that the vehicle dynamics and kinematics are symmetric, then, if the vehicle is in one of the states $\boldsymbol{x}_{\mathcal{T}(t)(\cdot)}$, the return trajectory provides an obstacle-free trajectory back to where it started. We will make this assumption about static environments and vehicle symmetry throughout the remainder of this chapter.

For feasibility reasons that will be explained shortly, the last state of $\mathcal{T}(t)$ is constrained to lie in an initial feasible invariant set. This places an extra condition on the set of feasible initial states for the trajectory planning problem at time 0. For a helicopter this could just be a hover state, for a UAV it could be a loiter circle which would be executed in the

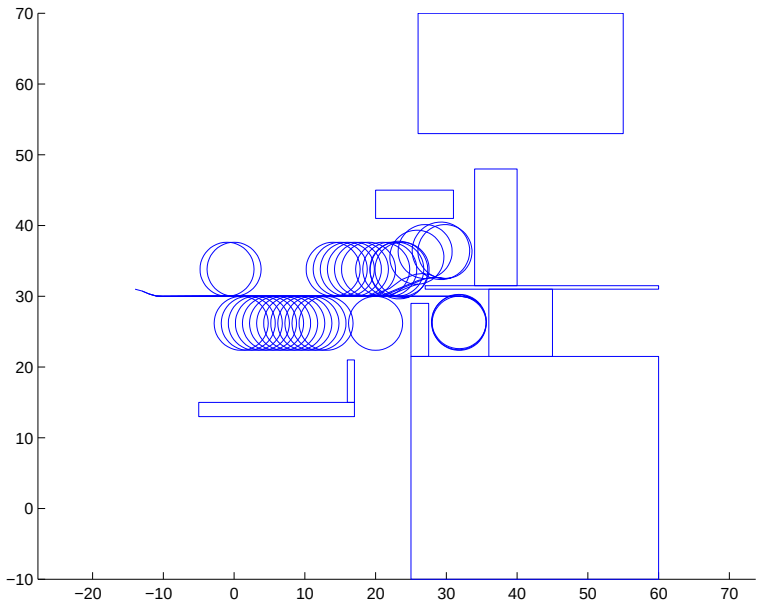


Figure 4-13: Intermediate receding horizon trajectories ending in loiter circles for the scenario in which the UAV gets trapped inside a concavity.

opposite direction if the vehicle decided to return. Next, we must ensure that the return trajectory is reachable from the terminal feasible invariant set, which is done using the following concept:

Definition 4.4 (Backtrack Pattern): *Define a* backtrack pattern $\mathcal{B}(t+T|t)$ *of length* $Q \leq T$ *as a sequence of* Q *dynamically feasible, obstacle-free states planned at time* t *that provides a transition from the terminal safe invariant set* $\mathcal{S}(t+T|t)$ *to the return trajectory* $\mathcal{T}(t)$*:* $\mathcal{B}(t+T|t) = \{\boldsymbol{x}_{\mathcal{B}(t)}(q) \in \mathbb{R}^n, q = 0, \ldots, Q \mid \boldsymbol{x}_{\mathcal{B}(t)}(q) \in \mathcal{X}(T), \boldsymbol{x}_{\mathcal{B}(t)}(q) \notin \mathcal{O}(t), \boldsymbol{x}_{\mathcal{B}(t)}(0) \in \mathcal{S}(t+T|t), \boldsymbol{x}_{\mathcal{B}(t)}(Q) \in \mathcal{T}(t), \forall q < Q\, \exists \boldsymbol{u}_{\mathcal{B}(t)}(q) \in \mathcal{U}(T-1) : \boldsymbol{x}_{\mathcal{B}(t)}(q+1) = \boldsymbol{A}\boldsymbol{x}_{\mathcal{B}(t)}(q) + \boldsymbol{B}\boldsymbol{u}_{\mathcal{B}(t)}(q)\}$.

The backtrack trajectory $\mathcal{B}(t+T|t)$ must be planned along with the terminal feasible invariant set $\mathcal{S}(t+T|t)$ during the receding horizon iteration at time t. If both are part of the optimization problem, the vehicle always has the option to either stay in the feasible invariant set (e.g., a loiter circle) or to backtrack along its path. Safety can then be defined as follows:

Definition 4.5 (Safety): *The vehicle is in a* safe state *at time* t *if that state lies on the return trajectory* $\mathcal{T}(t)$ *or if from that state there exists a dynamically feasible, obstacle-free trajectory of length* T *ending in a terminal feasible invariant set* $\mathcal{S}(t+T|t)$*, from which the vehicle has the possibility to return to* $\mathcal{T}(t)$ *along a backtrack pattern* $\mathcal{B}(t+T|t)$*.* Safety *is then defined as being in such a safe state at all times.*

Safety and the concept of a terminal feasible invariant set, a backtrack pattern and the return trajectory are illustrated in Figure 4-14.

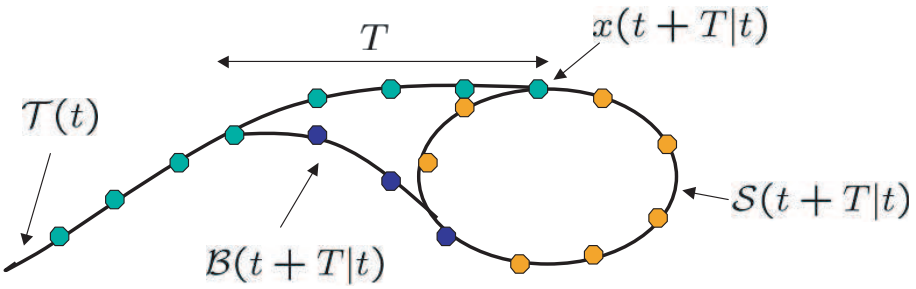


Figure 4-14: Safe receding horizon trajectory of T time steps ending in a terminal feasible invariant set $\mathcal{S}(t+T|t)$ and with backtrack pattern $\mathcal{B}(t+T|t)$ to the return trajectory $\mathcal{T}(t)$.

4.5.2 Safe Feasible Receding Horizon Planning Problem

Using the concepts defined above, we can now extend the feasible receding horizon trajectory planning problem (4.1)-(4.7) to account for safety. Consider the following formulation:

$$J_T^* = \min \sum_{k=0}^{T-1} \ell_k(\boldsymbol{x}(t+k|t), \boldsymbol{u}(t+k|t), \boldsymbol{x}_f) + f_T(\boldsymbol{x}(t+T|t), \boldsymbol{x}_f) \tag{4.15}$$

subject to:

$$\begin{aligned}
\boldsymbol{x}(t+k+1|t) &= \boldsymbol{A}\boldsymbol{x}(t+k|t) + \boldsymbol{B}\boldsymbol{u}(t+k|t), \\
&\quad k = 0, \dots, T-1 && (4.16)\\
\boldsymbol{x}(t|t) &= \hat{\boldsymbol{x}}(t|t-1) && (4.17)\\
\boldsymbol{x}(t+k|t) &\in \mathcal{X}(k) \supseteq \mathcal{X}(k+1), \\
&\quad k = 1, \dots, T && (4.18)\\
\boldsymbol{u}(t+k|t) &\in \mathcal{U}(k) \supseteq \mathcal{U}(k+1), \\
&\quad k = 0, \dots, T-1 && (4.19)\\
\boldsymbol{p}(t+k|t) &\notin \mathcal{O}_a(t),\ k = 1, \dots, T && (4.20)
\end{aligned}$$

$$\begin{cases} \boldsymbol{x}(t+T|t) \in \mathcal{S}(t+T|t) \\ \exists s \in \{0, \dots, I_p - 1\} : \boldsymbol{x}_{\mathcal{S}(t)}(t+T+s|t) \in \mathcal{B}(t+T|t) \end{cases} \quad \text{AND} \tag{4.21}$$

$$\{\boldsymbol{x}(t+T|t) \in \mathcal{T}(t) \quad \text{OR} \tag{4.22}$$

where I_p is the period of the terminal feasible invariant set $\mathcal{S}(t+T|t)$ of interest. Constraints (4.21) express that the vehicle must have the option to leave $\mathcal{S}(t+T|t)$ and transition to the backtrack pattern $\mathcal{B}(t+T|t)$ to eventually arrive on the return trajectory $\mathcal{T}(t)$. Either the combination of the two constraints (4.21) 'OR' the single constraint (4.22) must be satisfied to guarantee safety of the vehicle at all future time steps. Constraint (4.22) is necessary to maintain feasibility of the problem when the vehicle is already on the return trajectory. Since it may not be able to fit terminal feasible invariant sets in the environment when it is returning, the return trajectory then still provides the vehicle with a feasible solution. We denote the above trajectory optimization problem by $\mathrm{P}_S(\boldsymbol{x}(t), \mathcal{X}(k), \mathcal{U}(k), \mathcal{O}(t), \mathcal{S}(t+T|t), \mathcal{B}(t+T|t), \mathcal{T}(t))$.

Lemma 4.1: *Feasibility of the trajectory optimization problem* $\mathrm{P}_S(\boldsymbol{x}(t), \mathcal{X}(k), \mathcal{U}(k), \mathcal{O}(t), \mathcal{S}(t+T|t), \mathcal{B}(t+T|t), \mathcal{T}(t))$ *at time* t *implies that* $\boldsymbol{x}(t)$ *is a safe state.*

Proof: By construction of $\mathrm{P}_S(\cdot)$ and the definitions of $\mathcal{S}(t+T|t)$, $\mathcal{B}(t+T|t)$, and $\mathcal{T}(t)$, the existence of a feasible solution implies that the requirements for safety (as stated in Definition 4.5) of the initial state $\boldsymbol{x}(t)$ are satisfied. □

Theorem 4.2: *If the initial trajectory optimization problem* $\mathrm{P}_S(\boldsymbol{x}(0), \mathcal{X}(k), \mathcal{U}(k), \mathcal{O}(0), \mathcal{S}(T|0), \mathcal{B}(T|0), \mathcal{T}(0))$ *at time* $t = 0$ *with* $\mathcal{T}(0) = [\boldsymbol{p}'(0) \ -\boldsymbol{v}'(0)]' \in \mathcal{S}(0)$ *is feasible, then* $\mathrm{P}_S(\boldsymbol{x}(t), \mathcal{X}(k), \mathcal{U}(k), \mathcal{O}(t), \mathcal{S}(t+T|t), \mathcal{B}(t+T|t), \mathcal{T}(t))$ *will have a feasible solution at all future time steps* t*.*

Proof: We will show that if a feasible solution to the optimization problem exists at time step t, we can construct a feasible solution for the problem at time step $t+1$. Then, by induction, if the problem is feasible at $t = 0$, it will be feasible at all future time steps. Depending on the nature of the solution at t, we distinguish between three cases that span the space of all alternatives:

1. The planned final state $\boldsymbol{x}(t+T|t)$ lies in $\mathcal{S}(t+T|t)$ and there exists a $s^*(t) \geq 1$ such that $\boldsymbol{x}_{\mathcal{S}(t)}(t+T+s^*(t)|t) \in \mathcal{B}(t+T|t)$. The solution to $\mathrm{P}_S(\boldsymbol{x}(t), \mathcal{X}(k), \mathcal{U}(k), \mathcal{O}(t), \mathcal{S}(t+T|t), \mathcal{B}(t+T|t), \mathcal{T}(t))$ is then given by a sequence of states $\boldsymbol{x}(t+k|t),\ k = 0, \ldots, T$ where $\boldsymbol{x}(t+T|t) \in \mathcal{S}(t+T|t)$. Let a proposed solution to $\mathrm{P}_S(\boldsymbol{x}(t+1), \mathcal{X}(k), \mathcal{U}(k), \mathcal{O}(t+1), \mathcal{S}(t+T+1|t), \mathcal{B}(t+T+1|t+1), \mathcal{T}(t+1))$ be formed by $\boldsymbol{x}(t+1+k|t+1) = \boldsymbol{x}(t+k+1|t), k = 0, \ldots, T-1$ and $\boldsymbol{x}(t+1+T|t+1) = \boldsymbol{x}_{\mathcal{S}(t)}(t+T+1|t) \in \mathcal{S}(t+T|t)$. This proposed solution thus consists of the remaining states of the previous trajectory with an additional state in the terminal feasible invariant set $\mathcal{S}(t+T|t)$ computed at time t. Such a state exists per definition of a feasible invariant set. Hence we can set $\mathcal{S}(t+1+T|t+1) = \mathcal{S}(t+T|t)$. Furthermore, because the subsequent constraint sets $\mathcal{X}(k)$ and $\mathcal{U}(k)$ are supersets of respectively $\mathcal{X}(k+1)$ and $\mathcal{U}(k+1)$, the proposed trajectory still satisfies the kino-dynamic constraints. Also, because the new trajectory is the remaining part of the previous one and ends in the previous terminal feasible invariant set $\mathcal{S}(t+T|t)$, it lies within the environment as it was characterized at t, such that new obstacles in $\mathcal{O}_a(t+1)$ will not come into play and the avoidance constraints are still satisfied. Lastly, there will exist a step $s^*(t+1)$ at which the vehicle can transition from $\mathcal{S}(t+1+T|t+1) = \mathcal{S}(t+T|t)$ to $\mathcal{B}(t+1+T|t+1) = \mathcal{B}(t+T|t)$, namely $s^*(t+1) = s^*(t) - 1 \geq 0$. Furthermore, because no new obstacles come into play, $\mathcal{B}(t+T|t)$ is still a feasible backtrack pattern at $t+1$. The proposed trajectory thus satisfies all constraints of the problem at $t+1$ and is therefore a feasible solution.

2. The planned final state $\boldsymbol{x}(t+T|t)$ lies in $\mathcal{S}(t+T|t)$ and $\boldsymbol{x}_{\mathcal{S}(t)}(t+T|t) \in \mathcal{B}(t+T|t)$ with $s^*(t) = 0$. The reasoning is the same as before, except that now $s^*(t+1) = I_p - 1$. Indeed, since $\mathcal{S}(t+1+T|t+1) = \mathcal{S}(t+T|t)$ is periodic, there exists a step $s^*(t+1) = I_p - 1$ such that $\boldsymbol{x}_{\mathcal{S}(t+1)}(t+1+T+I_p-1|t+1) = \boldsymbol{x}_{\mathcal{S}(t)}(t+T|t) \in \mathcal{B}(t+T|t) = \mathcal{B}(t+1+T|t+1)$. A feasible solution exists again.

3. The planned final state $\boldsymbol{x}(t+T|t)$ lies on $\mathcal{T}(t)$, joining the return trajectory at a state r^*: $\boldsymbol{x}(t+T|t) = \boldsymbol{x}_{\mathcal{T}(t)}(r^*)$. The proposed solution is again the same as in case 1, except for the last state which is now: $\boldsymbol{x}(t+1+T|t+1) = \boldsymbol{x}_{\mathcal{T}(t)}(r^*+1) \in \mathcal{T}(t) \subset \mathcal{T}(t+1)$. By definition of $\mathcal{T}(t)$, this state is a feasible transition from $\boldsymbol{x}(t+T|t) = \boldsymbol{x}_{\mathcal{T}(t)}(r^*)$. Moreover, since $\mathcal{T}(t) \subset \mathcal{T}(t+1)$, the proposed final state also lies in $\mathcal{T}(t+1)$. Hence, a feasible solution to the trajectory optimization problem $\mathrm{P}_S(\boldsymbol{x}(t+1), \mathcal{X}(k), \mathcal{U}(k), \mathcal{O}(t+1), \mathcal{S}(t+T+$

$1|t), \mathcal{B}(t+T+1|t+1), \mathcal{T}(t+1))$ exists in this case as well. □

Corollary 4.2: *By Lemma 4.1 and Theorem 4.1, if a feasible solution to the initial trajectory optimization problem* $\mathrm{P}_S(\boldsymbol{x}(0), \mathcal{X}(k), \mathcal{U}(k), \mathcal{O}(0), \mathcal{S}(T|0), \mathcal{B}(T|0), \mathcal{T}(0))$ *at time* $t = 0$ *exists, safety will be maintained at all times.*

Remarks

1. Notice that the theorem does not provide performance or stability guarantees in the sense of reaching the goal. We are indeed mainly interested in a problem formulation that provides feasibility and safety guarantees as a basis before tackling stability. As discussed in the introduction, the latter is primarily related to the cost-to-go function. The optimization problem might indeed keep the vehicle trapped in a local minimum corresponding to staying in the safe invariant set. Furthermore, since at the start of a mission, the environment is not characterized beyond a certain distance and new obstacles are detected as the vehicle maneuvers through the environment, the vehicle may only realize along the way that the goal might actually not be reachable. In both cases, a higher level decision making algorithm can then decide to let the vehicle transition to its backtrack pattern and return trajectory or to incorporate new information about destination and obstacles in the cost function. This could simply consist of a change of $\boldsymbol{p}_f$ to $\boldsymbol{p}(0)$, i.e., the position where the vehicle started. The constraints of the optimization will remain the same, however. Regardless of the actual cost function, a safe trajectory back to the starting point will always exist.
2. The fact that a feasible solution is known *a priori* can be exploited when an optimal or feasible solution cannot be found within the timing constraints of real-time guidance system. The vehicle can then resort to the a priori solution as a safe backup plan. Moreover, the a priori solution can be used as a warm start for the optimization routine.
3. We assumed that the environment is static, such that the parts of the environment that were characterized when the vehicle maneuvered through it do not change. This assumption was needed to maintain an obstacle-free return trajectory $\mathcal{T}(t)$. However, if the environment is dynamic beyond the current detection region, the vehicle still has the terminal feasible invariant set $\mathcal{S}(t+T|t)$ to resort to. More generally, if the constraints $\exists s \in \{0, \ldots, S_{max}\} : \boldsymbol{x}_{\mathcal{S}(t)}(t+T+s|t) \in \mathcal{B}(t+T|t)$ and $\boldsymbol{x}(t+T|t) \in \mathcal{T}(t)$ are removed, we get the original formulation from Section 4.3 again, for which feasibility can still be proved: the vehicle can always remain in $\mathcal{S}(t+T|t)$. However, safety according to Definition 4.5 does not necessarily hold then: as shown in the scenario of Figure 4-12, the vehicle may get trapped in $\mathcal{S}(t+T|t)$. If we are only interested in maintaining feasibility through a partially-known environment, however, problem formulation (4.1)-(4.7) suffices. We will call this "minimal safety" and use it for the distributed algorithm in Chapter 5.
4. In practice, the OR-constraint in the optimization could be decided upon by a higher level planning unit before the next actual receding horizon iteration. Indeed, in case the vehicle is on its way back, the terminal safe invariant set and backtrack pattern constraints are not strictly necessary: the vehicle knows it has a safe way back to its starting point. On the other hand, when still moving towards the goal point, the vehicle should be able to fit its terminal feasible invariant sets in the environment. In that case, the $\boldsymbol{x}(t+T|t) \in \mathcal{T}(t)$ constraint can be removed. Again, the vehicle already knows that it could fly backwards along a feasible backtrack pattern. The return trajectory constraint is only needed to maintain feasibility of the same problem formulation in case the vehicle cannot fit terminal feasible invariant sets in the environment on its way back.

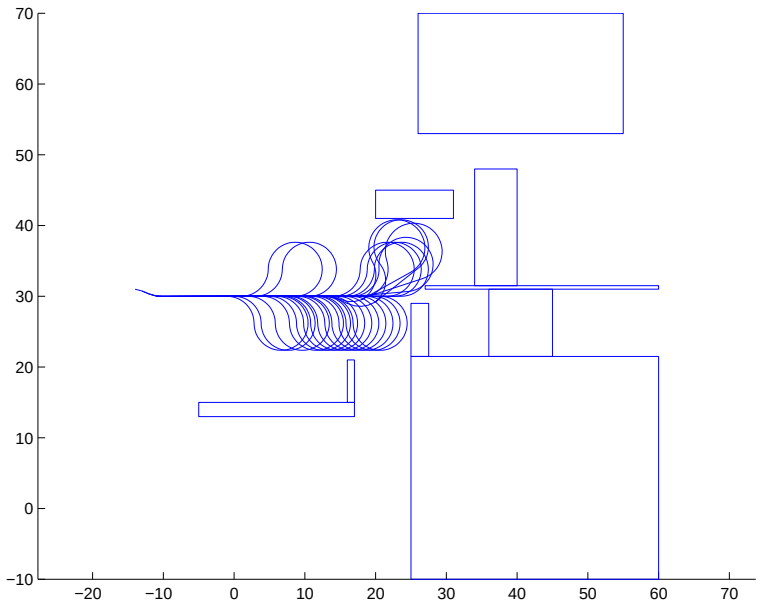


Figure 4-15: Intermediate receding horizon trajectories with backtrack pattern. The vehicle now avoids entering and getting trapped in the concavity because the loiter circle with additional backtrack pattern do not fit inside of it.

4.5.3 Backtrack Pattern

Similarly to terminal feasible invariant sets, states along the backtrack pattern could be expressed as affine transformations of the final state in the planning horizon. In that case, the backtrack pattern will have a fixed geometry and the sample points must correspond to actual time steps. Namely, the timing along the backtrack trajectory $\mathcal{B}(t+T|t)$ must be such that the vehicle leaves the feasible invariant set $\mathcal{B}(t+T|t)$ at an exact time step s^* and reaches the return trajectory $\mathcal{T}(t)$ at an exact state $\boldsymbol{x}_{\mathcal{T}(t)}(t^*)$. If the vehicle is flying at a constant speed, this can for example be accomplished by using a backtrack pattern consisting of a quarter circle starting at 270 deg along the loiter circle, with the same radius but reversed turning direction. For the feasible but unsafe scenario from Figure 4-12, this is shown in Figure 4-15. The vehicle can now not fit the loiter circle and backtrack trajectory inside the concavity and safely avoids it as a result. Note however, that the vehicle now gets steered towards the concavity above it, and must incorporate information about the detected geometry in the cost function to get out of it. However, thanks to the safety constraints, a feasible path back always exists.

More generally, the geometry of the backtrack trajectory and the state $\boldsymbol{x}_{\mathcal{T}(t)}(t^*)$ where it joins the return trajectory $\mathcal{T}(t)$ do not have to be fixed. The backtrack requirements can just consist of the kino-dynamic and obstacle avoidance constraints together with a fixed departure state from the terminal feasible invariant set and a selection of return states $\boldsymbol{x}_{\mathcal{T}(t)}(\cdot)$. All that is required then is a constraint stating that some state along the backtrack trajectory must join the return trajectory in at least one of the given states $\boldsymbol{x}_{\mathcal{T}(t)}(\cdot)$. This formulation is more flexible but also requires more binary variables. A result for the previous scenario is plotted in Figure 4-16. It can indeed be seen that the shape of the backtrack trajectory changes along the trajectory.

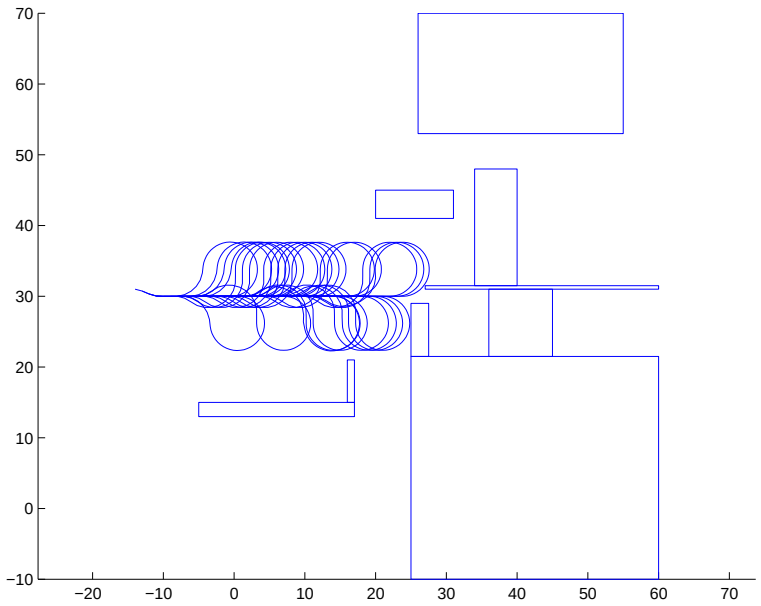


Figure 4-16: Intermediate receding horizon trajectories with backtrack patterns that are not geometrically constrained and can join the return trajectory in any one state of a given subset.

4.5.4 Safe Receding Horizon Planning without Feasibility Guarantees

If one is not interested in maintaining feasibility during all the receding horizon iterations, but only in guaranteeing safety of the vehicle, a practical approach is to simply constrain the last state of the planning horizon to be the start and end points of a dynamically feasible pattern with fixed shape. Sample positions along that pattern can again be expressed as affine transformations of $\boldsymbol{x}(t+T|t)$, either as actual time steps or as regular geometric points. No invariance property of the pattern is required, which results in loss of feasibility, but not of safety. Even if the optimization at the next iteration becomes infeasible, the vehicle still has a safe trajectory available that it can execute and that will bring it back to where it came from.

A scenario of such setup is given in Figures 4-17 and 4-18. The safety pattern again consists of the 270 deg part of a circle with a reverse quarter circle attached to it. However, to ensure that the start and end points of the safety pattern coincide, an additional line segment is required of length $2\alpha_c||\boldsymbol{v}(T)||$, i.e., twice the radius of the loiter circle with entry speed $||\boldsymbol{v}(T)||$. The line segment should be placed between the last state $\boldsymbol{x}(t+T|t)$ of the planning horizon and the first point on the return arcs. Notice that the pattern scales again with the entry velocity, adapting the latter to the available space in the environment. The red triangles along the trajectory in 4-17 indicate positions where the optimization became infeasible: the fixed geometric pattern could not be fitted at the end of the trajectory at the next step. However, the vehicle could just keep moving along the previous solution. As a result, feasibility was lost for several time steps, but safety was maintained, eventually guiding the vehicle past the infeasibilities.

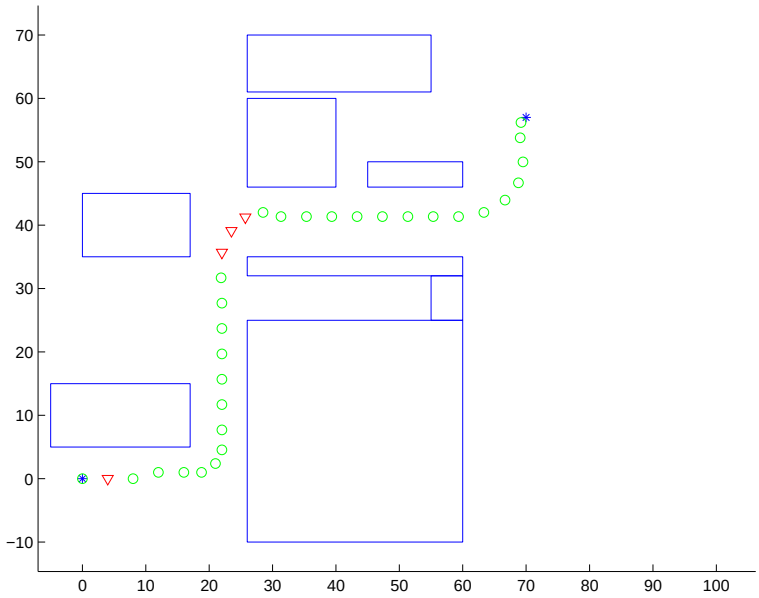


Figure 4-17: Safe receding horizon trajectory without feasibility guarantees. The red triangles indicate positions where the problem became infeasible (but safety was maintained).

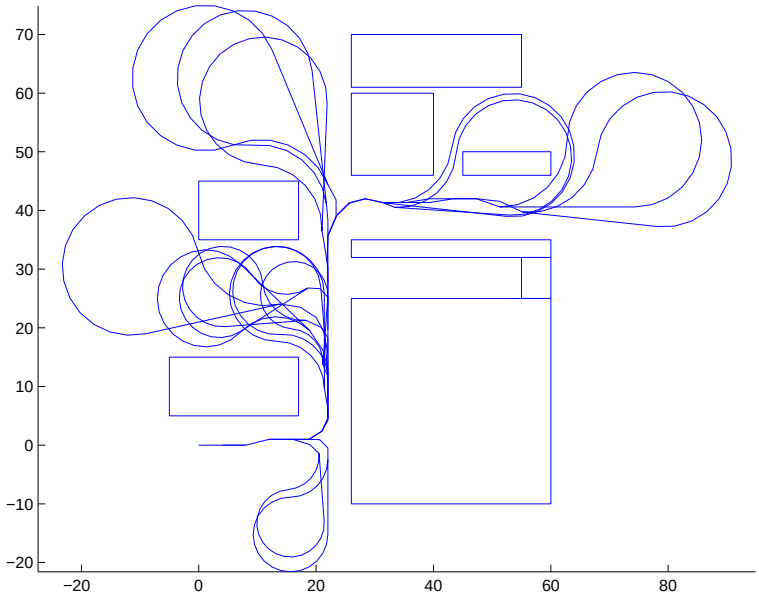


Figure 4-18: Intermediate receding horizon trajectories with safety patterns that do not contain a terminal invariant set.

4.6 Conclusion

This chapter extended the basic receding trajectory optimization problem to account for feasibility and safety constraints. Feasibility of the optimization problem at all time steps was ensured by constraining the intermediate receding horizon trajectories to terminate in feasible invariant sets. These sets can be expressed as affine transformations on the last state in the horizon and as such are planned online as part of the optimization problem. Examples of a fixed-wing UAV using left and right turning loiter constraints were given. Next, safety was ensured by maintaining feasible backtrack patterns to a return trajectory that will bring the vehicle back to where it started. A safe receding horizon formulation with feasibility guarantees was then presented for which simulation results were given. Finally, an alternative safe formulation without feasibility guarantees that is computationally more efficient was discussed.

Chapter 5

Safe Distributed Trajectory Planning for Multiple Vehicles

This chapter presents an algorithm for provably safe distributed trajectory planning of multiple autonomous vehicles that builds on the feasibility concepts developed in Chapter 4. Each vehicle plans its trajectory individually using a receding horizon strategy. Safety is guaranteed by maintaining, at each time step, dynamically feasible trajectories for all vehicles that terminate in distinct feasible invariant sets. Conflicts between multiple vehicles are resolved in a sequential, distributed cooperative fashion, in which each vehicle takes into account the latest trajectory and feasible invariant set of the other vehicles. Besides maintaining feasibility, the approach also provides an a priori safe rescue solution if the problem is too complex to be solved within the time constraints of a real-time system. The algorithm is applied to the case of multiple aircraft using loiter patterns as feasible invariant sets. Several examples of conflict situations resolved by the proposed method are presented.

5.1 Introduction

The growing complexity of global air traffic has highlighted the shortcomings of the current air traffic control infrastructure. In the current system, air traffic controllers use predefined routes and standard procedures to ensure safe separation between the aircraft in their sector. By doing so, maintaining safety remains a manageable problem for the controller; however, the routes followed by the individual aircraft are often suboptimal. For example, the system disallows the aircraft to fly directly to the destination or to take advantage of favorable winds [147]. With the air space becoming more congested, delays are more frequent and accidents caused by the air traffic controller more likely to occur.

Therefore, in recent years, the concept of *Free Flight* has emerged, which allows pilots to choose their own routes, altitude and speed. Safe conflict detection and resolution schemes constitute the basis of such a system, and have been a topic of active research. Automating these procedures reduces the risk of human errors and allows for optimization of the individual aircraft trajectories. Both *noncooperative* and *cooperative* conflict resolution methods have been proposed. In the noncooperative case, the aircraft involved in the conflict do not exchange information on their intentions and do not trust one another. Hence, a worst-case approach is adopted. Examples include the work of Tomlin *et al.*, in which a game-theoretic method is outlined [147, 146]. Safe protocol-based maneuvers are derived by precomputing reachability sets that account for the uncertainty in the actions of

the other aircraft [96, 4]. Although, theoretically, the method can be applied to any number of aircraft, the computational requirements for more than three aircraft become prohibitive.

In cooperative conflict resolution schemes, the aircraft do exchange information on their positions and intentions. Within this class of methods, one can further distinguish between *centralized* and *distributed* techniques, depending on whether the conflict is resolved by a central supervising controller or by each aircraft individually. In the former case, the position of each aircraft is known to the central controller who designs the individual trajectories for all aircraft, typically by solving one large optimization problem. Several centralized methods with hard anti-collision constraints have been proposed, including approaches based on semidefinite programming [37], nonlinear programming [108], mixed-integer linear programming [127, 114, 103], mixed-integer nonlinear programming [102], and variational analysis [53]. However, since the number of inter-vehicle combinations in the planning problem increases polynomially as $n(n-1)/2$ with the number of aircraft n involved in the conflict, the computation times of these methods typically scale exponentially.

By using a receding horizon approach, where the problem is solved over a limited time horizon that is shifted forward at each iteration, the computation time can be decreased. However, as discussed in Chapter 4, unless the problem is solved to completion at each iteration – which defeats the purpose of using a receding horizon–, safety is not guaranteed. Namely, the algorithms may fail to provide a solution in future time steps, due to aircraft that are located beyond the surveillance and planning radius of the aircraft accounted for at the current time step. For instance, when the planning horizon is too short and the maximum turn rate relatively small, the aircraft might approach one another too closely before accounting for each other in their plans. As a result, they might not be able to turn away in time, which would translate into the optimization problem becoming infeasible.

The scaling problem is also apparent in the field of unmanned aerial vehicles. In applications where path planning and coordination of a large fleet of autonomous vehicles is required, centralized solutions quickly become computationally intractable. Moreover, in these applications, the planning problem typically needs to be resolved multiple times, as new information about the environment is often gathered while the mission unfolds. Thus, a *distributed receding horizon* planning strategy seems a natural approach to solving the multi-vehicle trajectory generation problem. One such method is proposed in [137], where static obstacles and other moving agents are accounted for by potential functions. Although computationally attractive, the use of potential functions does not guarantee safety: obstacles and other vehicles are captured using soft "constraints" in the cost function. In [55], an alternative algorithm based on an iterative bargaining scheme is given. However, as the iteration might converge to an infeasible equilibrium, again only soft safety guarantees exist.

In this chapter, we present a model predictive control framework for distributed, *non-iterative* cooperative path planning of multiple aircraft with *hard* safety constraints. The novelty lies in the fact that safety is guaranteed by explicitly computing and maintaining a safe trajectory for each aircraft, without having to precompute an invariant set. Moreover, safety can be guaranteed for any number of interacting vehicles. Using the feasible invariant set principle introduced in Chapter 4, this is achieved by ensuring *a priori* that each aircraft can always transition to a dynamically feasible loiter pattern that is computed online. These loiter patterns act as safe backup paths in case no better solution to a conflict can be found in time. Conflicts are resolved in a sequential, distributed fashion in which each aircraft takes into account the latest trajectory and loiter pattern of the other when updating its own path.

Although we are primarily interested in the air traffic control application and the coordination problem of multiple fixed-wing UAVs, we present a safe distributed algorithm for autonomous vehicles in general. Afterwards, it is applied to the case of multiple aircraft and MILP is used to compute the trajectories. Note, however, that the algorithm will be formulated independently of the actual trajectory planning method and as such can be used with other optimization frameworks as well.

The chapter is organized as follows. Section 5.2 presents the problem formulation and Section 5.3 provides a high level description of the algorithm. Section 5.4 then gives the detailed MILP formulation with the aircraft model and the loiter and avoidance constraints. Next, Section 5.5 applies the framework to some example scenarios.

5.2 Problem Formulation

5.2.1 Receding Horizon Planning

The general problem tackled in this chapter is that of computing optimal trajectories for a set of (unmanned) vehicles while guaranteeing safety at all times. Each vehicle individually computes its trajectory towards a destination waypoint, accounting for the intentions of the other ones. Since information on the other vehicles is gathered online and changes as they update their trajectories, each vehicle adopts a receding horizon planning strategy. We again assume that the destination of each vehicle i consists of a final position $\boldsymbol{p}_{i,f}$ and a corresponding speed vector $\boldsymbol{v}_{i,f}$ with respect to an inertial coordinate frame. As before, they constitute the final waypoint or state $\boldsymbol{x}_{i,f} = [\boldsymbol{p}'_{i,f}\ \boldsymbol{v}'_{i,f}]'$.

We will denote the sequence of $T+1$ states and T inputs resulting from solving the path planning problem for vehicle i at a certain time step t respectively as $\mathbf{x}_i(t)$ and $\mathbf{u}_i(t)$:

$$\mathbf{x}_i(t) = \begin{bmatrix} \boldsymbol{x}_i(t|t) \\ \boldsymbol{x}_i(t+1|t) \\ \vdots \\ \boldsymbol{x}_i(t+T|t) \end{bmatrix},\ \mathbf{u}_i(t) = \begin{bmatrix} \boldsymbol{u}_i(t|t) \\ \boldsymbol{u}_i(t+1|t) \\ \vdots \\ \boldsymbol{u}_{i,}(t+T-1|t) \end{bmatrix},\ \mathbf{p}_i(t) = \begin{bmatrix} \boldsymbol{p}_i(t|t) \\ \boldsymbol{p}_i(t+1|t) \\ \vdots \\ \boldsymbol{p}_i(t+T|t) \end{bmatrix},$$

where – for notational convenience later on– we denote the position part of the trajectory separately as $\mathbf{p}_i(t)$.

As discussed in Chapter 2, the trajectory starting at time step t must be computed during time step $t-1$, i.e., when the vehicle is on its way to $\boldsymbol{x}_i(t|t)$. This state is part of the previous plan, which, as before, we assume to be accurately tracked. As such, the vehicle will be in $\boldsymbol{x}_i(t|t)$ when the next plan is executed. This implies that all vehicles can reliably assume that all others are exactly following their trajectories as planned. Including robustness to uncertainties in the latter is a topic of current research [70].

In what follows, we will denote the optimization problem for vehicle i that computes the trajectory starting at time step t as $\mathcal{M}_i(t)$. It accounts for the (predicted) initial state $\boldsymbol{x}_i(t|t)$, the destination waypoint $\boldsymbol{x}_{i,f}$, and constraints $\boldsymbol{x}_i(t+k|t) \in \mathcal{X}(k)$ on the states and $\boldsymbol{u}_i(t+k|t) \in \mathcal{U}(k)$ on the input commands. We will call a solution *acceptable* if it is feasible and its cost lies within a predefined optimality gap.

5.2.2 Safety Principle

As was discussed in Chapter 4, feasibility for a single vehicle i can be guaranteed by constraining the intermediate trajectory $\mathbf{x}_i(t)$ computed at each time step t to terminate in a feasible invariant set $\mathcal{S}_i(t+T|t)$ that lies outside of any forbidden zones in the environment. As such, at the next time step $t+1$, a feasible solution to the trajectory optimization problem is always available *a priori*, namely, the remaining part of the previous trajectory $\mathbf{x}_i(t)$ with an extra time step in $\mathcal{S}_i(t+T|t)$. Hence, if at time step $t+1$ no *acceptable* solution to the trajectory optimization problem can be found within the timing limits of a hard real-time guidance system, the previous trajectory can be followed as a safe backup plan. If necessary, the latter can be tracked all the way from time step t to $t+T$, thus arriving at the ingress state $\boldsymbol{x}_i(t+T|t)$ of the terminal feasible invariant set $\mathcal{S}_i(t+T|t)$, in which the vehicle can remain for an indefinite period of time. In this chapter, with a slight abuse of terminology, we will use the term "safety" for *minimal safety*, defined as having only a terminal feasible invariant set constraint to maintain feasibility, i.e., we will not account for the backtrack pattern and return trajectory:

Definition 5.1 (Minimal Safety): *We say that a vehicle is in a* minimally safe state *at time t if from that state, there exists a dynamically feasible, obstacle-free trajectory of T states ending in a terminal feasible invariant set $\mathcal{S}(t+T|t)$.* Minimal safety *is then defined as being in such a minimally safe state at all times.*

5.2.3 Conflict Description

The principle of maintaining a reachable feasible invariant set is key to ensuring (minimal) safety in case of an encounter between multiple vehicles. It will form the basis of our distributed algorithm for safe trajectory planning and conflict resolution. For simplicity of exposition, we will assume that the planning horizons of all vehicles are of equal length same. Extending the algorithm to the more general case of unequal planning horizons can be done at the cost of a more complicated notation. We start with some definitions:

Definition 5.2 (Conflict Zone): *We denote by $\mathcal{R}_i(t) \subset \mathbb{R}^n$ the subset of the inertial space with dimension n that encompasses the area in which all dynamically feasible trajectories lie that start at $\boldsymbol{x}_i(t|t) = [\boldsymbol{p}_i'(t|t)\ \boldsymbol{v}_i'(t|t)]'$ and end in terminal feasible invariant sets*: $\forall \mathbf{u}_i(t) \in \mathcal{U} = \{\mathcal{U}(0), \ldots, \mathcal{U}(T-1)\} : \mathbf{p}_i(t) \cup \mathcal{S}_i(t+T|t) \subset \mathcal{R}_i(t)$. *We call $\mathcal{R}_i(t)$ the* conflict zone *of vehicle i at time step t.*

As an initial condition (at $t=0$) for the planning and conflict resolution algorithm, we now assume that none of the conflict zones $\mathcal{R}_i(0)$ of the individual vehicles i ($i = 1, \ldots, K$) overlap. As such, at their initial positions $\boldsymbol{p}_i(0|0)$, all vehicles can safely plan their individual trajectories and terminal feasible invariant sets without accounting for the other vehicles.

Assumption 5.1 (Initial Safety): *At t=0, we have $\mathcal{R}_1(0) \bigcap \mathcal{R}_2(0) \ldots \bigcap \mathcal{R}_K(0) = \emptyset$.*

When the conflict zones of two or more vehicles start to overlap, however, the individually planned trajectories may intersect and lead to a collision. Therefore, as soon as an overlap is detected, the corresponding vehicles should account for each other's trajectories when updating their plans. We call this a *conflict*, and define it more formally as follows:

Definition 5.3 (Conflict): *We say that vehicle i is involved in a* conflict $\mathcal{C}_{ij}(t)$ *with vehicle $j \neq i$ at time step t, if $\mathcal{R}_i(t) \cap \mathcal{R}_j(t) \neq \emptyset$.*

Since a vehicle i computes the trajectory that starts at time step t during step $t-1$, $\mathcal{R}_i(t)$ needs to be determined at $t-1$ as well, based on the prediction of $\boldsymbol{x}_i(t|t)$. To denote the set of vehicles that are then involved in a conflict with vehicle i at time step t and for which collision avoidance constraints must be formulated, we introduce the following notation:

Definition 5.4 (Avoidance Set): *We denote by $\mathcal{J}_i(t) \subset \{1,\dots,K\}$ the subset of all vehicles $j \in \{1,\dots,K\}, j \neq i$, for which $\mathcal{R}_i(t) \cap \mathcal{R}_j(t) \neq \emptyset$. We call $\mathcal{J}_i(t)$ the* avoidance set *of vehicle i at time step t.*

If two vehicles i and j are involved in a conflict $\mathcal{C}_{ij}(t)$, however, their avoidance sets $\mathcal{J}_i(t)$ and $\mathcal{J}_j(t)$ are not necessarily the same: vehicle j can be involved in a conflict $\mathcal{C}_{jk}(t)$ with another vehicle k, but $\mathcal{R}_i(t) \cap \mathcal{R}_k(t) = \emptyset$. Hence, to maintain safety, vehicle j must account for both i and k, whereas vehicle i only has to account for j. However, although they are not directly in conflict, depending on the order in which the conflict is solved, the trajectory of i may still be influenced by that of k through the effect that k has on the trajectory of j. We therefore introduce the following set:

Definition 5.5 (Conflict Set): *We denote by $\mathcal{D}_i(t) \subset \{1,\dots,K\}$ the subset of all vehicles $j \in \{1,\dots,K\}$ for which there exists a vehicle sequence $k_1,\dots,k_S$ such that $\mathcal{R}_i(t) \cap \mathcal{R}_{k_1}(t) \neq \emptyset$, $\mathcal{R}_{k_1}(t) \cap \mathcal{R}_{k_2}(t) \neq \emptyset, \dots, \mathcal{R}_{k_D}(t) \cap \mathcal{R}_j(t) \neq \emptyset$. We call $\mathcal{D}_i(t)$ the* conflict set *of vehicle i at time step t.*

Note that if $j \in \mathcal{D}_i(t)$, we have $\mathcal{D}_j(t) = \mathcal{D}_i(t)$. Hence, we can define one set $\mathcal{D}(t) \equiv \mathcal{D}_i(t)$ that groups all vehicles that are connected through a chain of conflicts $\mathcal{C}_{ik_1}(t), \dots, \mathcal{C}_{k_Dj}(t)$ at time step t. Any vehicles that are not do not belong to $\mathcal{D}(t)$ will not come into play in resolving the conflict involving the vehicles in $\mathcal{D}(t)$. Hence, without loss of generality, we can restrict ourselves to solving the case of a single conflict $\mathcal{C}(t)$ associated with $\mathcal{D}(t)$. Other conflict sets may exist in the environment that can all be considered independent of each other.

By considering the vehicles in $\mathcal{D}(t)$ as vertices that are connected by an edge if a conflict $\mathcal{C}_{ij}(t)$ between a pair (i,j) exists, a graph can be associated with the conflict set $\mathcal{D}(t)$ that represents the inter-vehicle dependence at time step t. We define this graph as the *conflict graph* $\mathcal{G}(t)$:

Definition 5.6 (Conflict Graph): *We denote by $\mathcal{G}(t)$ the connected graph associated with conflict set $\mathcal{D}(t)$ at time step t that results from considering vehicles as vertices and edges as follows: a pair of vertices (i,j) is connected if there exists a conflict $\mathcal{C}_{ij}(t)$ between vehicles i and j at time step t.*

Independent conflict sets thus correspond to disconnected conflict graphs. Note that because the vehicles are moving, the conflict sets and associated conflict graphs may change at each time step. The planning algorithm proposed in this chapter will automatically account for this.

5.2.4 Communication Requirements

To determine the structure of the conflict graph there has to exist a communication network between the vehicles in the conflict sets. One possible communication strategy is to have two vehicles communicate with each other as soon as they detect each other within their conflict zones. Alternatively, all vehicles only communicate with a central hub (either a

ground station or leader vehicle) that keeps track of the positions $\boldsymbol{p}_i(t)$ and conflict zones $\mathcal{R}_i(t)$ of all vehicles i at all time steps t. Using this information it can determine the conflict set $\mathcal{D}(t)$ and corresponding conflict graph $\mathcal{G}(t)$, and communicate the avoidance sets $\mathcal{J}_i(t)$ back to the respective vehicles i. An additional benefit is that it can maintain the clocks of all vehicles synchronized, which will be crucial in the conflict resolution part of the algorithm.

Assumption 5.2 (Central Hub): *We assume that a central hub $\mathcal{H}$ is present with which all vehicles $i = 1, \ldots, K$ communicate. It keeps the clocks of all vehicles synchronized, and determines the avoidance sets $\mathcal{J}_i(t)$, the conflict set $\mathcal{D}(t)$, and the corresponding conflict graph $\mathcal{G}(t)$.*

Although the existence of a central hub is not crucial to our safety framework, it simplifies the required communication infrastructure and the presentation of the algorithm. The actual trajectory optimization will still be done in a distributed fashion, regardless of how information is exchanged between the vehicles. Decentralizing the communication itself is an ad hoc networking problem for which multiple techniques exist [121, 145].

To avoid collisions, vehicle i needs to account for the position sequence $\mathbf{p}_j(t)$ of all vehicles $j \in \mathcal{J}_i(t)$ when solving its trajectory optimization problem $\mathcal{M}_i(t)$. Moreover, to maintain future feasibility, its terminal feasible invariant set $\mathcal{S}_i(t+T|t)$ should not intersect with any of the terminal feasible invariant sets $\mathcal{S}_j(t+T|t)$ of the vehicles $j \in \mathcal{J}_i(t)$. For MILP-based trajectory planning, it suffices to describe all sets $\mathcal{S}_j(t+T|t)$ by the coordinates of a surrounding rectangular box which will then be considered a forbidden zone by vehicle i.

To solve its trajectory optimization problem $\mathcal{M}_i(t)$, vehicle i thus needs to obtain the latest trajectory information of all vehicles $j \in \mathcal{J}_i(t)$, and vice versa, the latter need to know the trajectory of i. Hence, either communication links between all vehicles in $\mathcal{J}_i(t)$ and vehicle i need to be set up, or the information can be distributed via the central hub. We now introduce the following notation to capture this trajectory information:

Definition 5.7 (Plan): *We denote by $\mathcal{P}_i(t)$ the sequence $\mathbf{p}_i(t)$ of trajectory points starting at time step t and the coordinates of a rectangular box that surrounds the terminal feasible invariant set $\mathcal{S}_i(t+T|t)$ of vehicle i. We call $\mathcal{P}_i(t)$ the* plan *of vehicle i at time step t.*

The plans $\mathcal{P}_j(t)$ for all vehicles $j \in \mathcal{J}_i(t)$ can then be included as avoidance constraints in the trajectory optimization problem $\mathcal{M}_i(t)$. We will denote this as the problem $\mathcal{M}_i(t)$ *s.t.* $\mathcal{P}_j(t)$, $\forall j \in \mathcal{J}_i(t)$.

A key step in our conflict resolution algorithm is that the vehicles in $\mathcal{D}(t)$ need to decide on an order $\mathcal{O}(t)$ in which each one updates its trajectory at time step $t-1$. An important property of the conflict graph $\mathcal{G}(t)$ is that vehicles that are not in direct conflict with each other, i.e., vehicle vertices that have no edge that directly connects them, can compute their trajectories simultaneously without affecting each other's feasibility. Indeed, since their conflict zones at the current time step do not overlap, their updated trajectories will not intersect.

In our algorithm, distinct subsets of such unconnected vehicle nodes will be allocated sequential time slots during which all vehicles of a particular subset optimize their individual trajectories simultaneously. To allocate as much computation time as possible to each vehicle, we therefore need to determine the minimum number of such unconnected vehicle subsets in $\mathcal{D}(t)$. This corresponds to a *vertex coloring problem* for the conflict graph $\mathcal{G}(t)$.

The goal is to assign colors $c(i)$ to each vertex i of $\mathcal{G}(t)$ such that no two vertices of the same color are directly connected while minimizing the number of colors for the entire graph. Brute-force algorithms could be used to find a minimum coloring, but Brelazs heuristic algorithm provides a good solution in faster time [23]. We will not elaborate further on such algorithms – see [70] for more details,– and will assume that the vehicles or the central hub can determine an appropriate coloring of the conflict graph $\mathcal{G}(t)$ and corresponding partitioning of the conflict set $\mathcal{D}(t)$ in subsets $\mathcal{F}_c(t)$. Instead, we will focus on the timing and safety properties of the trajectory planning algorithm. We summarize as follows:

Definition 5.8 (Conflict Subsets): *We denote by $\mathcal{F}_c(t), c = 1, \ldots, N_{\mathcal{G}(t)}$ the non-intersecting subsets of the conflict set $\mathcal{D}(t)$ resulting from solving a vertex coloring problem of the corresponding conflict graph $\mathcal{G}(t)$ at time step t. We call these subsets* conflict subsets. *In the minimal case, the number of conflict subsets $N_{\mathcal{G}(t)}$ is the chromatic number of the graph.*

Once a coloring and conflict partitioning have been determined, a computation order and corresponding time slots need to be assigned to the various conflict subsets. At each time step t, we therefore introduce time slots of length $\Delta t_{comp}(t) = \Delta t / N_{\mathcal{G}(t)}$ where Δt is the discretization step of the receding horizon. In our algorithm, each vehicle will have at least a time $\Delta t_{comp}(t)$ available to solve its trajectory planning problem. Notice, however, that some vehicles can make use of more time slots without taking time away from other vehicles. For example, vehicles i whose avoidance sets $\mathcal{J}_i(t)$ contain only vehicle vertices j of the same color can use the rest of the time step to optimize their trajectories after vehicles j used the previous time slot. Hence, all vehicles of a particular conflict subset $\mathcal{F}_c(t)$ will thus start optimizing at the same time, but are not necessarily allocated the same number of time slots. All have at least one slot of length $\Delta t_{comp}(t)$ available, however.

The previous insight leads us to the conclusion that there must exist an optimal order among the conflict subsets that maximizes the number of vehicles that can be allocated *more* than one time slot while ensuring that all vehicles get *at least* one slot. A good heuristic is to start with the conflict subset that is the most connected, i.e., the one for which the union of the avoidance sets of its members is the largest. Note, however, that this is not necessarily the largest conflict subset. We again assume that the central hub determines this order of the vehicle subsets and assigns the start and number of time slots to each individual vehicle:

Assumption 5.3 (Conflict Order): *We assume that the central hub $\mathcal{H}$ can determine an order $\mathcal{O}(t) = \{ord(c), c = 1, \ldots, N_{\mathcal{G}(t)}\}$ among the conflict subsets $\mathcal{F}_c(t)$, and a corresponding sequence of non-overlapping time slots $\{[t_{ord(c),s}, t_{ord(c),f} + \Delta t_{comm}), c = 1, \ldots, N_{\mathcal{G}(t)}\}$ that allocate the start time $t_{ord(c),s}$ and end time $t_{ord(c),f}$ during which the vehicles in the conflict set with order number $ord(c)$ should solve their trajectory optimization problems at time step $t-1$. Δt_{comm} is the time required to broadcast the plan $\mathcal{P}_i(t)$ of a vehicle $i \in \mathcal{F}_c(t)$ to the vehicles in its avoidance set $\mathcal{J}_i(t)$. We assume that this time is the same for all vehicles. If $\mathcal{J}_i(t)$ only contains vehicles j that have $ord(c(j)) < ord(c(i))$ then vehicle i can occupy the remainder of the time step and will be allocated a time slot $[t_{ord(c(i)),s}, (t-1-\Delta t_{comm}) + \Delta t_{comm})$. In both cases, we will denote the allocated time slot for vehicle i as $[t_{ord(c(i)),s}, t_{i,f} + \Delta t_{comm})$.*

Since a vehicle may leave the conflict set, or a new one may enter, the central hub must redetermine $\mathcal{D}(t)$ and $\mathcal{O}(t)$ at every time step. Using the preceding definitions, we can

now formulate a distributed optimization strategy for safe trajectory planning and conflict resolution.

5.3 Safe Trajectory Planning Algorithm

5.3.1 Algorithm

Starting at $t = 0$, at each following time step t, let all vehicles i in the environment $(i = 1, \ldots, K)$ execute the following planning algorithm:

Start: Start at time t.

- **Step 1**: Predict the *next state* $\boldsymbol{x}_i(t+1)$ and send it to the *central hub* $\mathcal{H}$. Next, receive the *avoidance set* $\mathcal{J}_i(t+1)$ with *conflict order numbers* $ord(c(j))$ of all vehicles $j \in \mathcal{J}_i(t+1)$ and the time slot $(t_{ord(c(i)),s}, t_{i,f})$ from $\mathcal{H}$.
- **Step 2a**: If $\mathcal{J}_i(t+1) \neq \emptyset$, broadcast the *current plan* $\mathcal{P}_i(t)$ to all vehicles in $\mathcal{J}_i(t+1)$ and go to **Step 3**. Else, go to **Step 2b**.
- **Step 2b**: Solve $\mathcal{M}_i(t+1)$. If an *acceptable solution* $\mathbf{x}_i(t+1) \cup \mathcal{S}_i(t+T+1|t+1)$ is found before time $t+1$, let the *new plan* $\mathcal{P}_{i,t+1} = \mathbf{p}_i(t+1) \cup \mathcal{S}_i(t+T+1|t+1)$. Else, let $\mathcal{P}_i(t+1) = \mathcal{P}_i(t) \setminus \boldsymbol{p}_i(t)$. Go to **End**.
- **Step 3**: At time $t_{ord(c(i)),s}$, solve $\mathcal{M}_i(t+1)$ *s.t.* $\mathcal{P}_j(t+1), j \in \mathcal{J}_i(t+1) : ord(c(j)) < ord(c(i))$ *and s.t.* $\mathcal{P}_k(t), k \in \mathcal{J}_i(t+1) : ord(c(k)) > ord(c(i))$.
- **Step 4**: If an *acceptable solution* $\mathbf{x}_i(t+1) \cup \mathcal{S}_i(t+T+1|t+1)$ is found at $t_{i,f}$, let the *new plan* $\mathcal{P}_i(t+1) = \mathbf{p}_i(t+1) \cup \mathcal{S}_i(t+T+1|t+1)$. Else, let $\mathcal{P}_i(t+1) = \mathcal{P}_i(t) \setminus \boldsymbol{p}_i(t)$.
- **Step 5**: During $(t_{i,f}, t_{i,f} + \Delta t_{comm})$, broadcast $\mathcal{P}_i(t+1)$ to all vehicles $k \in \mathcal{J}_i(t+1) : ord(c(k)) > ord(c(i))$.

End: End by time $t+1$, and repeat.

By construction, this algorithm maintains minimal safety for all vehicles: at each time step, there exists, *a priori*, a collision-free dynamically feasible trajectory for each vehicle. Namely, since all other vehicles during their last update accounted for the latest plan of the vehicle that is currently planning, the remaining part of its previous trajectory ending in a feasible invariant set can always be used as a safe "backup" plan. We formalize this in the following theorem:

Theorem 5.1: *Given a conflict-free situation at time $t = 0$ (Assumption 5.1), the above planning algorithm will maintain minimally safe trajectories (according to Definition 5.1) for all vehicles $i = 1, \ldots, K$ at all time steps.*

Proof: We prove the minimal safety property by using a double induction argument over the sequence of time steps and over the sequence of conflict subsets within a time step. By Assumption 1, at $t = 0$, all vehicles are in a minimally safe state (as defined in Definition 5.1) and $\bigcap_{i=1\ldots K} \mathcal{R}_i(0) = \emptyset$. Hence, by definition of $\mathcal{R}_i(0)$, all vehicles i have non-intersecting plans at $t = 0$. They thus all have an *a priori* minimally safe trajectory available at $t = 1$, namely, $\mathcal{P}_i(0) \setminus \boldsymbol{p}_i(0)$ ending in non-intersecting terminal feasible invariant sets $\mathcal{S}_i(t+T|t)$ at time step T, the length of the planning horizon.

Assume now that at time step $t = k$, all vehicles have safe plans $\mathcal{P}_i(k)$. Before computing its next plan $\mathcal{P}_i(k+1)$, a particular vehicle i will know whether its avoidance set $\mathcal{J}_i(k+1)$ is empty or not by communicating with the central hub $\mathcal{H}$ (Step 1). In case $\mathcal{J}_i(k+1) = \emptyset$, any feasible solution to $\mathcal{M}_i(k+1)$ will be minimally safe, since the conflict zone $\mathcal{R}_i(k+1)$ does not intersect with that of any other vehicles. If an acceptable solution cannot be found in time, $\mathcal{P}_i(k)\backslash \boldsymbol{p}_i(k)$, *i.e.* the previous plan excluding the current state, is still valid and available as a minimally safe backup plan at time step $k+1$.

In case $\mathcal{J}_i(k+1) \neq \emptyset$, vehicle i will obtain its order number $ord(c(i)) \in \{1 \dots N_{\mathcal{G}(k+1)}\}$ from the central hub $\mathcal{H}$. If $ord(c(i)) = 1$, vehicle i is the first to update its trajectory at time step k, along with all other vehicles in the conflict subset $\mathcal{F}_{c(i)}(k+1)$. Given that the plan $\mathcal{P}_i(k)$ was minimally safe, a minimally safe solution to $\mathcal{M}_i(k+1)$ *s.t.* $\mathcal{P}_j(k+1), \forall j \in \mathcal{J}_i(k+1)$ continues to exist, namely, $\mathcal{P}_i(k)\backslash \boldsymbol{p}_i(k)$. In the nominal case, $\mathcal{P}_i(k+1)$ will differ from $\mathcal{P}_i(k)\backslash \boldsymbol{p}_i(k)$. Since the updated plan $\mathcal{P}_i(k+1)$ is constrained to avoid the existing trajectories $\mathcal{P}_j(k), \forall j \in \mathcal{J}_i(k+1)$, all other vehicles in the conflict set $\mathcal{D}(k+1)$ remain in minimally safe states. The same reasoning holds for all other vehicles s in the conflict subset $\mathcal{F}_{c(i)}(k+1)$. Since, per definition of $\mathcal{F}_{c(i)}(k+1)$, their avoidance sets $\mathcal{J}_s(k+1)$ do not contain any of the other vehicles with $ord(c(s)) = 1 = ord(c(i))$, their plans will not affect each other.

Now, consider a vehicle i' with $ord(c(i')) > 1$, and assume that it has a minimally safe backup plan $\mathcal{P}_{i'}(k)\backslash \boldsymbol{p}_{i'}(k)$. It will account 1) for the previous plans $\mathcal{P}_{j'}(k)$ for all following vehicles $j' \in \mathcal{J}_{i'}(k+1) : ord(c(j')) > ord(c(i'))$ and 2) for the new plans $\mathcal{P}_{l'}(k+1)$ of all prior vehicles $l' \in \mathcal{J}_{i'}(k+1) : ord(c(l')) < ord(c(i'))$. Given that a minimally safe plan $\mathcal{P}_{i'}(k)\backslash \boldsymbol{p}_{i'}(k)$ for vehicle i' exists, the problem is feasible. Thus 1) all vehicles j' with $ord(c(j')) > ord(c(i'))$ will still have minimally safe backup plans $\mathcal{P}_{j'}(k)\backslash \boldsymbol{p}_{j'}(k)$ available when they update their paths, and 2) the new plans $\mathcal{P}_{l'}(k+1)$ of all vehicles l' with $ord(c(l')) < ord(c(i'))$ will remain minimally safe. Again, the same reasoning holds for all other vehicles s' in the conflict subset $\mathcal{F}_{c(i')}(k+1)$. Because their avoidance sets $\mathcal{J}_{s'}(k+1)$ do not contain any of the other vehicles with $ord(c(s')) = ord(c(i'))$, their plans will not affect each other.

Hence, by induction over the sequence of conflict subsets, minimal safety for all vehicles is maintained within time step k. As a result, each vehicle is in a minimally safe state at the start of the next time step $k+1$. Thus, using induction once more, given the safety assumption at $t = 0$, minimal safety for all vehicles is maintained over the sequence of time steps.

5.3.2 Remarks

1. Note that the algorithm is *not* a bargaining or convergence process: at any given time step, each vehicle contributes only once to the solution of the conflict. As presented here, the algorithm should cycle through the full conflict set $\mathcal{D}(t+1)$ before the next time step $t+1$, i.e., the time at which all vehicles reach the first state in their plans. However, the more vehicles that are involved in the conflict graph, the longer each vehicle will typically take to solve its trajectory optimization problem. On the other hand, the allocated computation time scales inversely with the number of conflict subsets (i.e., colors). Therefore, if necessary, for a particular cycle of the algorithm, the central hub can distribute the computation times over several time steps. If during this cycle of longer duration all other vehicles keep tracking the latest trajectories that they communicated, safety for all vehicles is maintained. This is only the case, however, if no new vehicles enter the conflict set during that longer cycle. Since one iteration of the conflict resolution algorithm is now spread over more than one time

step, this can be guaranteed by computing conflict sets and subsets for longer planning horizons in Step 1 of the algorithm. Alternatively, the central hub could command the new vehicle not to update its plan and to keep following its latest trajectory, while it communicates this latest plan to the other vehicles. The latter can then still include that plan as an avoidance constraint in their respective trajectory optimizations, without making the problem infeasible.

2. Worst-case, if many vehicles are involved in the conflict graph, and the problem becomes too complex to be solved within the timing constraints of the algorithm, the vehicles will enter their (non-intersecting) terminal feasible invariant sets and may get trapped in it. This could also happen if the problem has certain geometric symmetries. In both cases, however, the central hub can then decide on subsequent subsets of the conflict sets (including the possibility of one by one) and allocate as much time as needed to compute a way out of the feasible invariant sets or break the symmetry. Such live-lock situations and the associated instability of not reaching the goal are primarily related to the cost function. Here, however, we are only concerned about guaranteeing feasibility. Typically, feasibility needs to be ensured before stability can be tackled and is therefore a problem of interest by itself. Ensuring stability to build a provably stable algorithm is still a topic of ongoing research [70].

3. It is natural to assume that the vehicles in the first conflict subset in the conflict order are favored because they can update their trajectories first. However, in our simulations this was not necessarily the case: we generally found that most vehicles have to make equal efforts (in terms of deviating from their nominal trajectories towards their destination) to resolve the conflict. Depending on the geometry of the individual trajectories, the first vehicle could even be more constrained than the last one in the order sequence. Moreover, by randomly changing the conflict subset order at each time step, no single vehicle should have an *a priori* advantage over any other.

4. The algorithm is robust to communication failures. As soon as a broken communication link is detected, the vehicles involved should execute their backup plans: they should keep following the trajectory that was last broadcast and eventually enter their terminal feasible invariant sets. The other vehicles can then keep accounting for those same plans when updating their trajectories.

5. The formulation of the actual trajectory optimization problem should account for the discrete-time nature of the conflict resolution algorithm: its constraints must guarantee collision avoidance during the continuous transition between the discrete trajectory sample points.

5.4 Implementation Using MILP

Although the conflict resolution algorithm can be used with other trajectory optimization techniques, we use MILP again to formulate the avoidance constraints and express the requirements for the feasible invariant sets. In the following, the framework is applied to the case of multiple aircraft using the loiter patterns from Chapter 4 as terminal feasible invariant sets. To highlight the performance of the conflict resolution algorithm, we present 2D scenarios. However, the algorithm can readily be used with 3D problems as well. The aircraft would then have an extra degree of flexibility, namely a change in altitude, to plan non-intersecting loiter patterns.

5.4.1 Aircraft Model

For our simulations, we use a 2D version of the first-order reference velocity model (3.15) from Section 3.4 that captures the dynamics of an actual aircraft more accurately than the double integrator model used in Chapters 2 and 4. Let $\boldsymbol{p}_i(t) = [x_i(t)\ y_i(t)]'$ and $\boldsymbol{v}_i(t) = [\dot{x}_i(t)\ \dot{y}_i(t)]'$ denote the inertial position and velocity vector again. As in (3.15), let the control input to aircraft i be the inertial reference speed vector $\boldsymbol{u}_i(t) = [\dot{x}_{cmd,i}(t)\ \dot{y}_{cmd,i}(t)]'$. The 2D version of the velocity control model (3.15) then becomes:

$$\begin{array}{rcl} \ddot{x}_i(t) & = & -\frac{1}{\tau_i}\dot{x}_i(t) + \frac{k_i}{\tau_i}\dot{x}_{cmd,i}(t) \\ \ddot{y}_i(t) & = & -\frac{1}{\tau_i}\dot{y}_i(t) + \frac{k_i}{\tau_i}\dot{y}_{cmd,i}(t) \end{array} \tag{5.1}$$

in which τ_i is again a time constant and k_i is a gain. In state space form, $\dot{\boldsymbol{x}}_i(t) = \mathbf{A}_i\boldsymbol{x}_i(t) + \mathbf{B}_i\boldsymbol{u}_i(t)$, we obtain:

$$\begin{bmatrix} \dot{x}_i(t) \\ \dot{y}_i(t) \\ \ddot{x}_i(t) \\ \ddot{y}_i(t) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\frac{1}{\tau_i} & 0 \\ 0 & 0 & 0 & -\frac{1}{\tau_i} \end{bmatrix} \begin{bmatrix} x_i(t) \\ y_i(t) \\ \dot{x}_i(t) \\ \dot{y}_i(t) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{k_i}{\tau_i} & 0 \\ 0 & \frac{k_i}{\tau_i} \end{bmatrix} \begin{bmatrix} \dot{x}_{cmd,i}(t) \\ \dot{y}_{cmd,i}(t) \end{bmatrix} \tag{5.2}$$

Using the bilinear transform with a discretization step Δt, the discrete-time model becomes:

$$\begin{bmatrix} x_i(k+1) \\ y_i(k+1) \\ \dot{x}_i(k+1) \\ \dot{y}_i(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & \frac{\Delta t}{2}\left(1 + \frac{2\tau_i - \Delta t}{2\tau_i + \Delta t}\right) & 0 \\ 0 & 1 & 0 & \frac{\Delta t}{2}\left(1 + \frac{2\tau_i - \Delta t}{2\tau_i + \Delta t}\right) \\ 0 & 0 & \frac{2\tau_i - \Delta t}{2\tau_i + \Delta t} & 0 \\ 0 & 0 & 0 & \frac{2\tau - \Delta t}{2\tau_i + \Delta t} \end{bmatrix} \begin{bmatrix} x_i(k) \\ y_i(k) \\ \dot{x}_i(k) \\ \dot{y}_i(k) \end{bmatrix}$$
$$+ \begin{bmatrix} \frac{k_i(\Delta t)^2}{2\tau_i + \Delta t} & 0 \\ 0 & \frac{k_i(\Delta t)^2}{2\tau_i + \Delta t} \\ \frac{2k_i\Delta t}{2\tau_i + \Delta t} & 0 \\ 0 & \frac{2k_i\Delta t}{2\tau_i + \Delta t} \end{bmatrix} \begin{bmatrix} \dot{x}_{cmd,i}(k) \\ \dot{y}_{cmd,i}(k) \end{bmatrix} \tag{5.3}$$

The magnitudes of the reference and actual inertial velocity are again bounded by a minimum required speed $v_{min,i}$ and a maximum achievable speed $v_{max,i}$:

$$\begin{array}{c} v_{min,i} \leq ||\boldsymbol{v}_i(t)|| \leq v_{max,i} \\ v_{min,i} \leq ||\boldsymbol{u}_i(t)|| \leq v_{max,i} \end{array} \tag{5.4}$$

To capture limits on turn rate and differences in the lateral and longitudinal aircraft dynamics, we reuse constraints (3.20a-b):

$$\begin{array}{rcl} ||\boldsymbol{a}_i(t) - \alpha_i\frac{\boldsymbol{v}_i(t)}{v_i(0)}|| & \leq & \beta_i \\ ||\boldsymbol{a}_i(t) + \alpha_i\frac{\boldsymbol{v}_i(t)}{v_i(0)}|| & \leq & \beta_i \\ ||\boldsymbol{a}_i(t)|| & \leq & a_{lat,max,i} \end{array} \tag{5.5}$$

where α_i and β_i are given by expressions (3.21a-b). Remember that these inequalities approximate the geometric profile that encompasses all inertial acceleration vectors $\boldsymbol{a}_i = [\ddot{x}_i\ \ddot{y}_i]'$ that are dynamically feasible. For the linearized and discrete time version of con-

straints (5.4) and (5.5), we refer the reader back to Section 3.4.

5.4.2 Loiter Circles

As discussed in Section 5.2, the trajectory at each iteration is constrained to terminate in a feasible invariant set. Like the examples in Chapter 4, we choose this to be a right or left turning loiter circle (see Figure 4-3), where the turn direction will be decided upon by the optimization problem. Remember that the loiter pattern was described by equally spaced sample points $\boldsymbol{p}_R(\theta)$ and $\boldsymbol{p}_L(\theta)$ along the circles. By introducing a rotation matrix $\mathbf{R}(\theta)$, these points were expressed as affine transformations of the last state $\boldsymbol{x}_i(T) = [\boldsymbol{p}_i(T)' \ \boldsymbol{v}_i(T)']'$ in the planning horizon:

$$\boldsymbol{p}_{R,i}(\theta) = \boldsymbol{p}_i(T) + \alpha_{c,i}(\mathbf{R}(\theta) - \mathbf{I})\boldsymbol{v}_i^{\perp}(T) \tag{5.6}$$

$$\boldsymbol{p}_{L,i}(\theta) = \boldsymbol{p}_i(T) - \alpha_{c,i}(\mathbf{R}(\theta) - \mathbf{I})\boldsymbol{v}_i^{\perp}(T) \tag{5.7}$$

The state $\boldsymbol{x}_i(T)$ thus acts as an entry state to the loiter circles of which the radius scales linearly with the magnitude of the entry velocity $\boldsymbol{v}_i(T)$. As such, by slowing down or speeding up, the aircraft has the flexibility to adapt the size of its loiter circle to the state of the environment, i.e., to the trajectories and locations of the loiter patterns of the other aircraft. In the following, these geometric relations will be used to express the avoidance constraints.

5.4.3 Avoidance Constraints

We assume here for simplicity that the conflict zone $\mathcal{R}_i$ of all aircraft i is a circle of fixed radius r_{reach}, where r_{reach} is the maximum distance that can be traveled over the length of the planning horizon, including a margin for the loiter circle. Denoting the radius of the loiter circle when flying at v_{max} by r_{max}, a simple calculation yields $r_{reach} = \sqrt{(T\Delta t v_{max} + r_{max})^2 + 4r_{max}^2}$. Note that this is the most conservative assumption: in general, an aircraft might not be able to reach the full area covered by the circle, and the shape of the conflict zone will typically depend on the initial velocity $\boldsymbol{v}_i(0)$. Furthermore, the radius will be different depending on the length of the planning horizon and the velocity and turn rate bound.

The avoidance set $\mathcal{J}_i$ can then easily be determined: it consists of all aircraft that are within a distance of $2r_{reach}$ of aircraft i at time step t. In addition, Assumption 5.1 reduces to all aircraft being separated by a distance greater than $2r_{reach}$: $\forall i, j \in \{1, \ldots, K\}, i \neq j : \|\boldsymbol{p}_i(0) - \boldsymbol{p}_j(0)\| > 2r_{reach}$. As discussed in Steps 2a and 5 of the trajectory planning and conflict resolution algorithm, when the conflict set $\mathcal{D} \neq \emptyset$, the aircraft will communicate their plans $\mathcal{P}$ to one another. Since we assumed that all vehicles can accurately follow the planned trajectories, their waypoints and loiter circles can be considered as translating obstacles with known motion by the aircraft that is currently planning. These constraints are now specified in more detail.

Regular Trajectory Points

Each trajectory point $\boldsymbol{p}_j(k)$ of aircraft $j \in \mathcal{J}_i$ is considered an obstacle that is present at time step k in the planning horizon, if that point lies in the conflict zone $\mathcal{R}_i$ of the planning aircraft i, i.e., when $\boldsymbol{p}_j(k) \in \mathcal{R}_i$. Denote the set of time steps for which the latter holds as $\mathcal{T}_j \subseteq \mathcal{T} = [0, \ldots, T]$. Due to the discrete-time nature of the trajectories, the waypoints

are considered square obstacles of dimension $2d_s = 2(\max(v_{max}\Delta t, d_{safe}) + v_{max}\Delta t)$, where d_{safe} is the required safety distance around each aircraft. The lower left corner of the waypoint obstacle $\boldsymbol{p}_j(k) = (x_j(k), y_j(k))$ is then given by $(x_{min,jk}, y_{min,jk}) = (x_j(k) - d_s, y_j(k) - d_s)$, the upper right corner by $(x_{max,jk}, y_{max,jk}) = (x_j(k) + d_s, y_j(k) + d_s)$.

For the trajectory points $(x_i(k), y_i(k)),\ k \in \mathcal{T}_j$ – which include points on the loiter pattern if the aircraft is using its backup plan,– the required safety distance with all other aircraft is then guaranteed at all times during the planning horizon, if these points satisfy the following set of constraints:

$\forall j \in \mathcal{J}_i,\ \forall k \in \mathcal{T}_j$:

$$\begin{array}{rcrcl} x_i(k) & \leq & x_{min,jk} & + & Mb_{j1}(k) \\ -x_i(k) & \leq & -x_{max,jk} & + & Mb_{j2}(k) \\ y_i(k) & \leq & y_{min,jk} & + & Mb_{j3}(k) \\ -y_i(k) & \leq & -y_{max,jk} & + & Mb_{j4}(k) \\ \sum_{n=1}^{4} b_{jn}(k) & \leq & 3 & & \\ b_{jn}(k) & \in & \{0,1\} & & \end{array} \tag{5.8}$$

As in previous chapters, $b_{jn}(k)$ are binary variables and M is a sufficiently large positive number. The last constraint again ensures that at least one of the inequality constraints is active, thereby guaranteeing that the trajectory point $(x_i(k), y_i(k))$ lies outside the waypoint obstacles of the other aircraft.

Loiter Points

To ensure that the loiter circle of the planning aircraft i does not intersect with that of any of the aircraft $j \in \mathcal{J}_i$, we derive equivalent constraints for the sample points along the circle. Assume that there are $N_L = \lceil \frac{2\pi}{\theta_s} \rceil$ of these, where θ_s denotes the discretization angle. Let the loiter circle of each aircraft $j \in \mathcal{J}_i$ be contained within a square with lower left corner $(x_{min,jL}, y_{min,jL})$ and upper right corner $(x_{max,jL}, y_{max,jL})$. As discussed in Chapter 4, these dimensions include a safety boundary that accounts for the continuous segments of the loiter circle of aircraft i between its discrete sample points. The two corner points, together with the time step at which the loiter is initiated, is the only information that needs to be exchanged about the loiter pattern.

The square that needs to be avoided by the loiter points of the planning aircraft i can now be considered a static obstacle. By introducing a binary variable d that selects either the right or left circle, and by substituting expressions (5.6)-(5.7) for the sample points in the avoidance constraints (5.8), we obtain:

$\forall j \in \mathcal{J}_i,\ \forall l \in [1, \ldots, N_L]$:

$$\left\{ \begin{array}{rcrcl} x_i(T) - \alpha_c(\cos l\theta_s - 1)\,\dot{y}_i(T) - \alpha_c(\sin l\theta_s)\,\dot{x}_i(T) & \leq & x_{min,jL} + Mb_{jl1} + Md \\ -x_i(T) + \alpha_c(\cos l\theta_s - 1)\,\dot{y}_i(T) + \alpha_c(\sin l\theta_s)\,\dot{x}_i(T) & \leq & -x_{max,jL} + Mb_{jl2} + Md \\ y_i(T) - \alpha_c(\sin l\theta_s)\,\dot{y}_i(T) + \alpha_c(\cos l\theta_s - 1)\,\dot{x}_i(T) & \leq & y_{min,jL} + Mb_{jl3} + Md \\ -y_i(T) + \alpha_c(\sin l\theta_s)\,\dot{y}_i(T) - \alpha_c(\cos l\theta_s - 1)\,\dot{x}_i(T) & \leq & -y_{max,jL} + Mb_{jl4} + Md \end{array} \right. \tag{5.9}$$

$$\begin{cases} x_i(T) + \alpha_c(\cos l\theta_s - 1)\,\dot{y}_i(T) + \alpha_c(\sin l\theta_s)\,\dot{x}_i(T) & \leq & x_{min,jL} + Mb_{jl1} + M(1-d) \\ -x_i(T) - \alpha_c(\cos l\theta_s - 1)\,\dot{y}_i(T) - \alpha_c(\sin l\theta_s)\,\dot{x}_i(T) & \leq & -x_{max,jL} + Mb_{jl2} + M(1-d) \\ y_i(T) + \alpha_c(\sin l\theta_s)\,\dot{y}_i(T) - \alpha_c(\cos l\theta_s - 1)\,\dot{x}_i(T) & \leq & y_{min,jL} + Mb_{jl3} + M(1-d) \\ -y_i(T) - \alpha_c(\sin l\theta_s)\,\dot{y}_i(T) + \alpha_c(\cos l\theta_s - 1)\,\dot{x}_i(T) & \leq & -y_{max,jL} + Mb_{jl4} + M(1-d) \end{cases} \tag{5.10}$$

$$\begin{aligned} \sum_{n=1}^{4} b_{jln} &\leq 3 \\ b_{jln},\ d &\in \{0,1\} \end{aligned} \tag{5.11}$$

Here, the index l denotes the angle around the circle, and does not necessarily correspond to the exact position of the aircraft on the circle after l time steps. Satisfying the above constraints ensures that no two loiter circles intersect; together with constraints (5.8) and the backup plan principle, they guarantee (minimal) safety of each aircraft at all times.

5.4.4 Cost Function

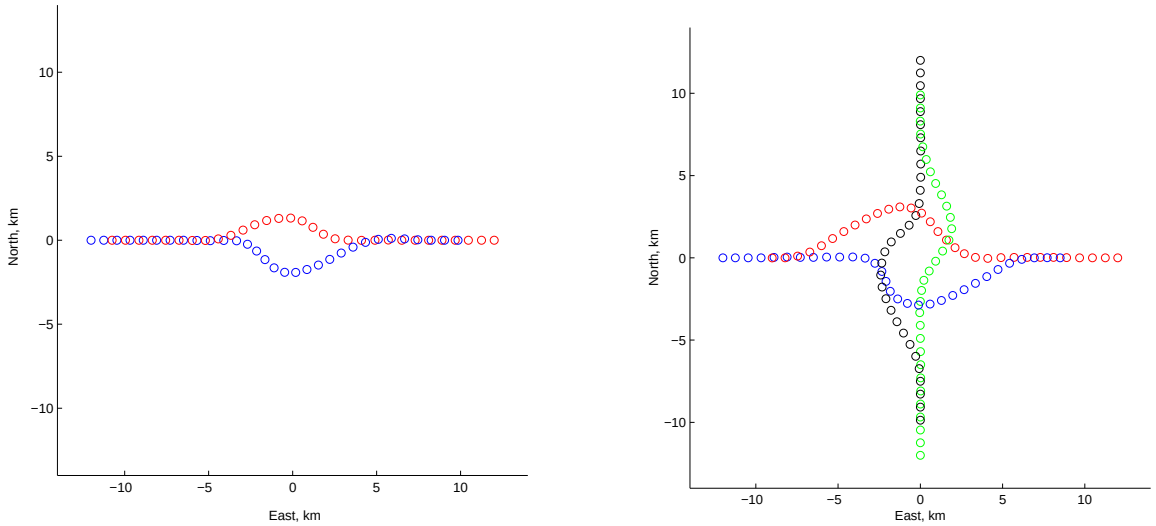
Combined with an appropriate (piecewise) linear cost function, the state space equations (5.3) together with the dynamic and kinematic constraints (3.25a)-(3.28), the avoidance and loiter constraints (5.8)-(5.11), constitute a MILP again that must be solved online at each iteration of the safe trajectory planning algorithm as outlined in Section 5.3. In our simulations, we used the following cost function for each aircraft i separately:

$$\min_{\boldsymbol{x}_i(k),\boldsymbol{u}_i(k)} J_T = \sum_{k=1}^{T} \left(\boldsymbol{q}' |\boldsymbol{x}_i(k) - \boldsymbol{x}_{f,i}| \right) - r(\boldsymbol{p}_{f,i} - \boldsymbol{p}_i(0))' \boldsymbol{v}_i(k) \tag{5.12}$$

It aims at proceeding towards the destination by i) minimizing the 1-norm of the difference with the desired state $\boldsymbol{x}_{f,i}$, and ii) maximizing the scalar product of the velocity vector with the vector $(\boldsymbol{p}_{f,i} - \boldsymbol{p}_i(0))$, indicating the direction from the initial position $\boldsymbol{p}_i(0)$ to the destination $\boldsymbol{p}_{f,i}$. Since the latter remains constant over the planning horizon, the scalar product is linear. Its effect is twofold: the vehicle will change its heading to aim straight for the goal while flying as fast as possible. As such, this cost function mimics a minimum time solution without requiring the desired state to be reachable within the planning horizon, as is the case for the exact shortest time formulation (see Chapter 3). As such, a significantly shorter planning horizon can be used, allowing for faster computation and efficient receding horizon planning over longer distances. Finally, $\boldsymbol{q}$ and r are weights that can be tuned appropriately.

5.5 Results

We now present some example scenarios to which the proposed trajectory planning algorithm was applied. The parameters of the aircraft used in our simulations were the following: $v_{max} = 160$ m/s, $v_{min} = 130$ m/s, $\tau = 5$ s, $k = 5$, $a_{fwd,max} = 15$ m/s^2 and $a_{lat,max} = 13.96$ m/s^2. The latter corresponds to a maximum turn rate of 5 deg/s. We simulated all trajectories for 30 to 65 iterations of the algorithm with a planning horizon of $T = 5$ time steps of $\Delta t = 5$ s each. The number of loiter sample points was set to $N_L = 8$, the number of linear inequalities to approximate circular constraints to $N = 16$.



(a) Safe trajectories of 2 aircraft.

(b) Safe trajectories of 4 aircraft.

Figure 5-1: Simulation results for distributed conflict resolution of 2 and 4 aircraft for 30 time steps of 5 s each.

For the avoidance constraints, we used a required safety distance of $d_{safe} = 1.5$ km for each aircraft, resulting in square waypoint obstacles of size $2d_s = 4.6$ km. The simulation results were again obtained on a Pentium 4 with 2.2 GHz clock speed, using the CPLEX 9.0 MILP solver [56] with an AMPL [34] and Matlab interface.

In the first scenario (see Figure 5-1a), two aircraft are flying in opposite directions and are bound to encounter each other in the origin. Aircraft 1 starts in position $(-12 \text{ km}, 0 \text{ km})$ flying west at 150 m/s, and is headed for a waypoint at $(12 \text{ km}, 0 \text{ km})$. Aircraft 2 does the opposite and starts at $(12 \text{ km}, 0 \text{ km})$, flying east at 150 m/s. When they encounter each other in the middle, they safely resolve the conflict by both turning right. Figure 5-2 shows the computation times of both aircraft at each iteration: with a maximum of 0.77 s, they are clearly within the timing constraints of the algorithm ($t_f - t_s < \Delta t/2 = 2.5$ s).

The second scenario (see Figure 5-1b) involves 4 aircraft. Aircraft 3 and 4 are now flying south and north at 150 m/s, starting in $(0 \text{ km}, 12 \text{ km})$ and $(0 \text{ km}, -12 \text{ km})$ respectively. Again, the conflict in the middle is safely resolved within the timing constraints of the algorithm (see Figure 5-3). Theoretically, each aircraft now has at most $\Delta t/4 = 1.25$ s available, but using a CPLEX computation time limit of 1.0 s, an acceptable solution was always found within 1.1 s. To illustrate the safety principle, Figure 5-4 displays the (non-intersecting) loiter boxes of the 4 aircraft for the first 9 time steps (corresponding to 45 s).

Figures 5-5 and 5-6 show similar scenarios, but now with respectively 8 and 10 aircraft starting on positions evenly distributed around circles of 12 and 24 km. In the 8 vehicle case, the aircraft have enough time to find a good feasible solution to the problem, whereas in the 10 vehicle case, some aircraft use their loiter circles as backup plans. From a conflict resolution point of view, it allows other vehicles to pass, whereas from a computational point of view, the circles provide a feasible trajectory in case no better solution can found in time. Even in these fairly complex scenarios, (minimal) safety for all aircraft is maintained at all times.

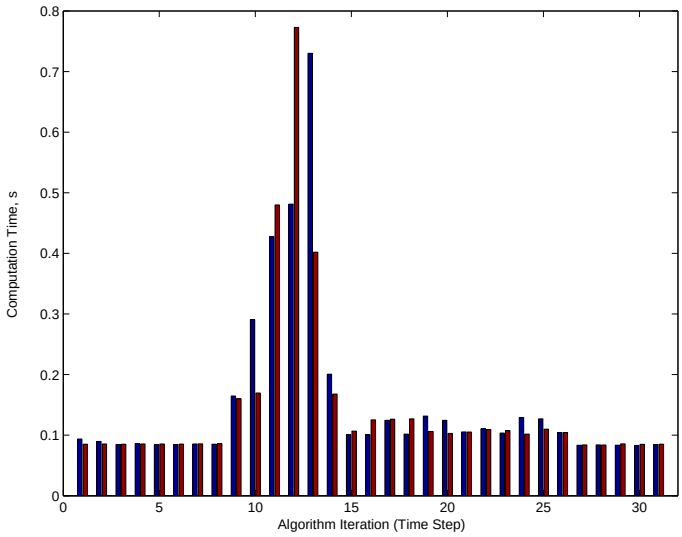


Figure 5-2: Computation times at each iteration for the 2 aircraft scenario of Figure 5-1a.

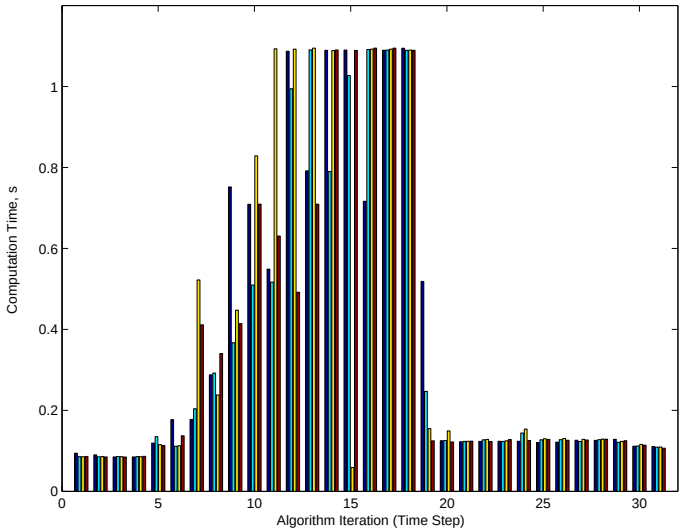


Figure 5-3: Computation times at each iteration for the 4 aircraft scenario of Figure 5-1b.

5.6 Conclusion

This chapter presented an algorithm for provably safe, distributed trajectory planning of multiple autonomous vehicles. A receding horizon strategy was adopted for each vehicle individually that accounts for the trajectories of the neighboring vehicles. Safety at all times was guaranteed by constraining the intermediate plans of all vehicles to terminate in feasible invariant sets that do not intersect. Conflicts between multiple vehicles were resolved in a distributed fashion while maintaining an *a priori* feasible backup plan for each vehicle. The vehicle interdependence and computation time allocation was determined as a graph coloring problem. The algorithm was implemented for multiple aircraft using loiter circles as terminal feasible invariant sets. Simulation scenarios with timing results were presented, illustrating the capability of our approach to resolve complex conflicts in real-time.

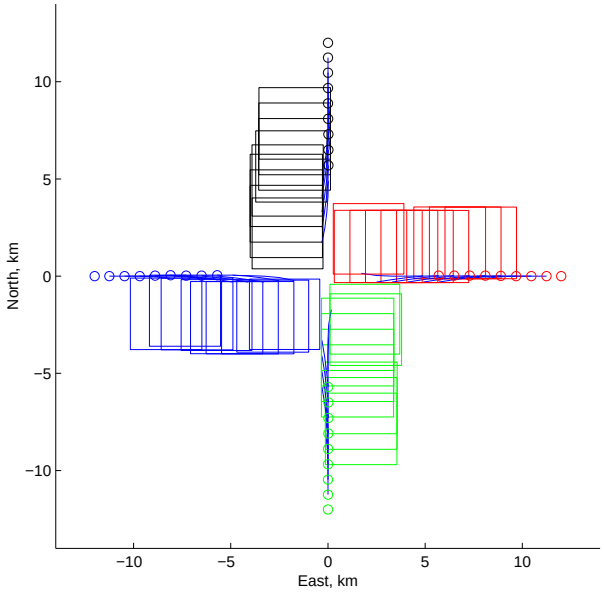


Figure 5-4: Safe trajectories and corresponding loiter boxes for the 4 aircraft scenario during the first 9 time steps (corresponding to 45 s).

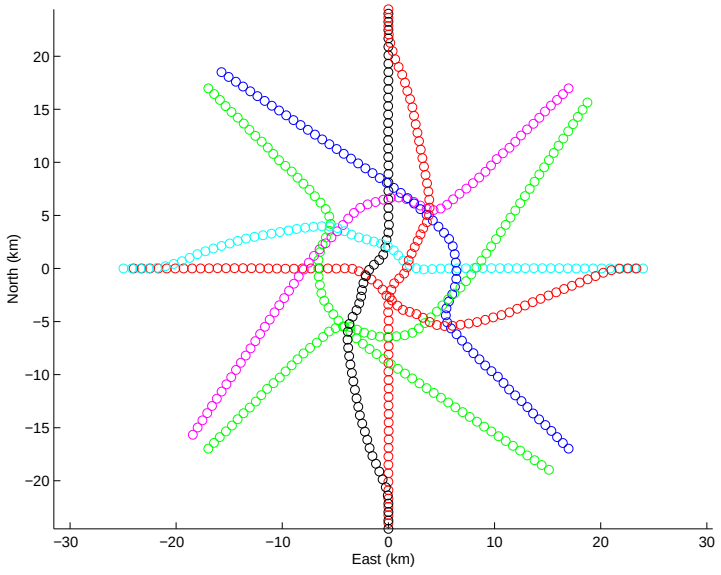


Figure 5-5: Safe trajectories of 8 aircraft during 40 time steps of 5 s each.

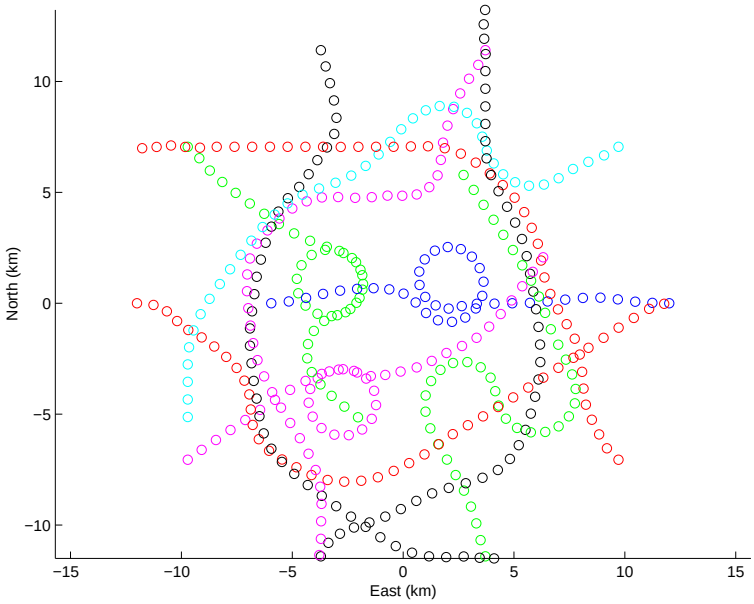


Figure 5-6: Safe trajectories of 10 aircraft during 65 time steps of 5 s each.

Chapter 6

Multi-Vehicle Path Planning for Non-Line of Sight Communication

This chapter presents a proof-of-concept application of the concepts developed in the previous chapters. We tackle the problem of online connectivity-constrained trajectory planning for autonomous helicopters through cluttered environments. A lead vehicle must execute a certain mission whereby wireless line of sight communication to its ground station is lost. Relay helicopters are therefore introduced that must position themselves in such way that indirect line of sight connectivity between the leader and the ground station is always maintained. This requires coordinated multi-vehicle trajectory optimization which is tackled using centralized and distributed receding horizon planning strategies. The problem is again formulated as a mixed-integer linear program that accounts for the vehicle dynamics, obstacle and collision avoidance, and connectivity constraints. A centralized two-helicopter mission is described for which simulation, hardware in the loop, and actual flight test results are presented. Simulation results for distributed scenarios with two and three vehicles are given as well.

6.1 Introduction

The problem of interest in this chapter is to guide a small autonomous helicopter through a cluttered (e.g. urban) environment in which line of sight communication with a ground station cannot be maintained due to the presence of obstacles. Depending on the nature of the mission, however, some form of communication with the vehicle might be required. For example, a ground operator may need the capability to upload mission level commands to the helicopter, such as to fly to and inspect a certain site, and request a real-time video downlink. In addition, he or she should have access to information about the state and health of the machine. Practical applications of interest are search and rescue missions, inspection of disaster sites, urban surveillance, traffic observation, volcanic crater measurements, fire fighting unit support, etc.

Depending on the power and antenna characteristics of the wireless transmitter/receiver with which the helicopter and ground station are equipped, obstruction of the line of sight by nearby buildings or natural obstacles such as hills might significantly degrade the reliability and quality of the communication link. Moreover, since the available power on-board the vehicle is limited because of battery life (or fuel, in case an alternator is used), the wireless range will naturally be constrained even in an obstacle-free environment. A reliable

communication link between the ground station and the helicopter executing the mission can then be obtained by introducing a set of relay agents. These can be a combination of ground and aerial vehicles, but we are interested in using only rotorcraft as relay stations: they can take advantage of altitude as an extra degree of freedom, and unlike fixed-wing aircraft, they can hover at a steady location. Indirect communication with the ground station can then be established by positioning the relay vehicles in such way that a sequence of direct line of sight links exists between the ground station and the mission helicopter, henceforth called the leader.

As the leader is maneuvering through the environment, the relay helicopters should adapt their positions such that visibility/connectivity between the subsequent relay agents is maintained. A cooperative scheme is therefore required that allows the relay formation to coordinate its reconfiguration with the actions of the leader, and, vice versa, the motion of the leader is constrained by the locations that the relay helicopters can reach. This coordination problem can be formulated as a connectivity-constrained multi-vehicle path planning problem, in which feasible trajectories for all agents are computed that optimize a certain cost criterion. Besides maintaining line of sight, feasibility of the trajectories implies respecting the helicopters' kino-dynamic constraints, as well as avoiding obstacles and collisions. We will focus solely on these dynamic and geometric aspects of the connectivity problem and ignore mobile networking factors such as fading, cross-talk, and delay, which can also affect the availability of links between the vehicles.

Various researchers have approached the problem of motion planning for connected multi-agent systems. Nguyen *et al.* [100] describe the use of mobile relay nodes to extend the effective range of a ground robot exploring a complex interior environment. The relay nodes follow the lead robot in convoy and automatically stop where needed to form an ad hoc network guaranteeing a link between the lead robot and the base station. Variations of this algorithm can be found in Sweeney *et al.* [143]. Another such incremental deployment strategy was developed by Howard *et al.* [50]. These approaches, however, are rule-based and as such are less flexible than the optimization-based framework presented in this chapter. In our methodology, a solution is determined online "from scratch" that accounts for the features of the particular environment and various constraints on the vehicles. Hybrid approaches in which predetermined navigation or communication recovery behaviors are dynamically employed online are described in Wagner and Arkin [150], Ulam and Arkin [148], and in Powers and Balch [107]. In the latter, the computational complexity for a planning agent is kept low by assuming that other team members remain in the same position one step into the future. Our distributed strategy uses a similar idea to maintain feasibility of the configuration during the trajectory updates of the planning agent in the relay sequence, more specifically by applying the terminal feasible invariant set principles from Chapters 4 and 5.

Next, Beard and McLain [5] use dynamic programming to tackle the problem of searching a region of interest using multiple UAVs under communication range and collision avoidance constraints. Obstacles blocking line of sight between vehicles are not accounted for, however. Various auction or so-called market-based algorithms have been developed as well, in which tightly coupled tasks are allocated in a distributed fashion and a solution is found through a convergence process. Examples include the work of Kalra *et al.* [59], Lemaire *et al.* [76] and Bererton *et al.* [15]. Their focus, however, is on the fair distribution of the tasks and workload among the agents rather than on the actual trajectory planning accounting for obstacle and collision avoidance. Finally, Spanos and Murray [141] discuss the feasibility aspects of path planning for vehicles connected through a range-constrained

wireless network. They define a connectivity robustness measure which quantifies the freedom of individual vehicles to undergo arbitrary motions without disconnecting the network, but do not explicitly construct feasible trajectories.

Unlike some of the approaches mentioned above, the focus of this chapter is not on the (positioning) task allocation, but on the development of an online cooperative trajectory planning algorithm that incorporates connectivity constraints directly into the formulation. In the most general case, different additional constraints can be placed on the individual vehicles: for example, one of the agents may have to remain at ground level, whereas the relay agents must stay above a certain altitude. Our framework will be able to naturally handle such requirements. More specifically, we will compute the connectivity-constrained trajectories online using MILP again with both centralized and distributed planning strategies. This allows us to apply the various concepts and algorithms developed in the previous chapters to a realistic proof-of-concept application and scenario.

6.2 Problem Formulation

6.2.1 Problem Setup

This section extends the basic centralized receding horizon formulation (2.3)-(2.9) for multiple vehicles as presented in Chapter 2 to account for connectivity constraints. Let the various agents be denoted by an index $i = 0, \ldots, L$, where $i = 0$ is the stationary ground station, $i = L$ indicates the leader, and the remaining index values correspond to the relay helicopters. The dynamics of the moving vehicles are again characterized by discrete-time, linear state space models $(\boldsymbol{A}_i, \boldsymbol{B}_i)$ with corresponding constraint sets $\mathcal{X}_i(t)$ and $\mathcal{U}_i(t)$. The obstacles that are relevant to vehicle i are again represented by the set $\mathcal{O}_i$.

The overall goal of the trajectory planning problem is for the lead vehicle L to perform a certain task while optimizing a particular cost. The leader's task is again specified as having to fly to a certain waypoint of interest, denoted as the final state $\boldsymbol{x}_f \equiv [\boldsymbol{p}_f' \ \boldsymbol{v}_f']'$, *e.g.*, to go take video footage at a particular location. As before, a series of task waypoints would be considered as a sequence of independent tasks. Besides maintaining feasibility w.r.t. obstacle and collision avoidance, the motion of the leader will be constrained by the network connectivity requirement and the communication capabilities of the relay helicopters. Denote the broadcasting range of each agent i as $d_{relay,i} \in \mathbb{R}$. Since we are considering two-directional communication between the vehicles, a communication link between two agents i and j can only exist if they are within each other's broadcasting range, or thus if the Euclidean distance $d_{ij} \equiv \|\boldsymbol{p}_i - \boldsymbol{p}_j\|$ between them is smaller than the shortest range of the two: $d_{ij} \leq \min(d_{relay,i}, d_{relay,j})$. Moreover, the visibility between them should not be obstructed. We define these requirements more precisely as follows:

Definition 6.1 (Line of Sight Connection): *We say that there exists a* line of sight connection *between two agents i and j, positioned respectively at $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$, if their Euclidean separation distance $d_{ij} = \|\boldsymbol{p}_i - \boldsymbol{p}_j\|$ is smaller than the shortest broadcasting range of the two, i.e., $d_{ij} \leq \min(d_{relay,i}, d_{relay,j})$, and the line between them does not intersect any obstacles, i.e., $\boldsymbol{p}_i + \lambda(\boldsymbol{p}_j - \boldsymbol{p}_i) \notin (\mathcal{O}_i \cap \mathcal{O}_j),\ 0 \leq \lambda \leq 1$.*

In the most general case, the topology of the wireless network could be set up ad hoc as part of the optimization problem. In this chapter, however, we consider a fixed order in the set of relay vehicles $i = 1, \ldots, L-1$ that corresponds to a linear graph structure: $i = 1$ relays information between $i = 0$ and $i = 2$, $i = 2$ does so between $i = 1$ and $i = 3$, and so

forth. As such, there is no robustness against link dropouts at one of the relay agents. At the cost of a more complex optimization problem (and therefore longer computation time), more flexible network structures with redundancy properties could be obtained. Such setup, however, would automatically limit the operational range of the vehicle platoon. Without redundancy and after having fixed the order of the relay vehicles, the maximum distance the leader could travel away from the ground station is given by : $\sum_{i=0}^{L-1} \min(d_{relay,i}, d_{relay,i+1})$, *i.e.*, in the best-case scenario in which no obstacles obstruct the line of sight between the leader and ground station.

The connectivity requirement for the ordered team of vehicles can then be defined as follows:

Definition 6.2 (System Connectivity): *We say that there is* system connectivity *in the ordered set of agents if there exist line of sight connections between all pairs of subsequent agents* $(i, i+1),\ i = 0, \ldots, L-1$: $d_{i(i+1)} \leq \min(d_{relay,i}, d_{relay,i+1})$ *and* $\boldsymbol{p}_i + \lambda(\boldsymbol{p}_{i+1} - \boldsymbol{p}_i) \notin (\mathcal{O}_i \cap \mathcal{O}_{i+1}),\ 0 \leq \lambda \leq 1$.

Trajectory feasibility for all vehicles now also implies that system connectivity is maintained at all times.

6.2.2 Centralized Receding Horizon Planning

For the same reasons as mentioned earlier in this thesis, namely computational complexity and/or partially unknown environments, we again make use of a receding horizon planning approach. We first present a centralized strategy, in which the ground station computes trajectories for all vehicles simultaneously. In Section 6.5, a distributed cooperative algorithm is presented, similar to the one discussed in Chapter 5.

The objective of the trajectory optimization problem is to now guide the leader to its goal state $\boldsymbol{x}_f$ while optimizing a certain cost and maintaining connectivity. Since the actions of the leader must be coordinated with those of the relay helicopters, the efforts made by the latter should somehow be reflected in the cost. We therefore introduce the following general objective function:

$$\begin{aligned} J_T &= \sum_{k=0}^{T-1} \ell_{L,k}(\boldsymbol{x}_L(t+k|t), \boldsymbol{u}_L(t+k|t), \boldsymbol{x}_f) + f_L(\boldsymbol{x}_L(t+T|t), \boldsymbol{x}_f) \\ &+ \sum_{i=1}^{L-1} \sum_{k=0}^{T-1} \ell_{i,k}(\boldsymbol{x}_i(t+k|t), \boldsymbol{u}_i(t+k|t)) \end{aligned} \tag{6.1}$$

in which $\ell_{L,k}(\cdot)$ indicates the stage cost associated with the leader at the k^{th} time step, and $f_L(\cdot)$ represents a terminal penalty function. As discussed earlier, the latter should be an estimate of the cost-to-go from the last state $\boldsymbol{x}_L(t+T|t)$ in the planning horizon to the desired waypoint $\boldsymbol{x}_f$. Similarly, $\ell_{i,k}(\cdot)$ is the k^{th} stage cost corresponding to relay vehicle $i,\ i = 1, \ldots, L-1$. Notice, however, that there are no cost-to-go functions associated with the relay vehicles and that their stage costs $\ell_{i,k}(\cdot)$ do not depend on $\boldsymbol{x}_f$. Indeed, reaching the goal waypoint only matters to the leader, and there are no *a priori* known desired locations for the relay helicopters.

The connectivity-constrained trajectory optimization problem at time t can then be

formulated as:

$$J_T^* = \min_{\boldsymbol{x}_i(\cdot),\boldsymbol{u}_i(\cdot)} \left\{ \sum_{k=0}^{T-1} \ell_{L,k}(\boldsymbol{x}_L(t+k|t), \boldsymbol{u}_L(t+k|t), \boldsymbol{x}_f) + f_L(\boldsymbol{x}_L(t+T|t), \boldsymbol{x}_f) + \sum_{i=1}^{L-1} \sum_{k=0}^{T-1} \ell_{i,k}(\boldsymbol{x}_i(t+k|t), \boldsymbol{u}_i(t+k|t)) \right\} \tag{6.2}$$

subject to: $\forall i = 1, \ldots, L:$

$$\begin{aligned}
\boldsymbol{x}_i(t+k+1|t) &= \boldsymbol{A}_i \boldsymbol{x}_i(t+k|t) + \boldsymbol{B}_i \boldsymbol{u}_i(t+k|t), \\
&\quad k = 0, \ldots, T-1 \\
\boldsymbol{x}_i(t|t) &= \hat{\boldsymbol{x}}_i(t|t-1) \\
\boldsymbol{x}_i(t+k|t) &\in \mathcal{X}_i(k),\ k = 1, \ldots, T \\
\boldsymbol{u}_i(t+k|t) &\in \mathcal{U}_i(k),\ k = 0, \ldots, T-1 \\
\boldsymbol{p}_i(t+k|t) &\notin \mathcal{O}_{a,i}(t),\ k = 1, \ldots, T \\
\|\boldsymbol{p}_i(t+k|t) - \boldsymbol{p}_j(t+k|t)\| &\geq d_{safe},\ j = 0,\ j \geq i+1,\ k = 1, \ldots, T \\
\boldsymbol{p}_i(t+k|t) + \lambda \left(\boldsymbol{p}_{i-1}(t+k|t) - \boldsymbol{p}_i(t+k|t)\right) &\notin \mathcal{O}_{v,i}(t) \cap \mathcal{O}_{v,i-1}(t),\ 0 \leq \lambda \leq 1, \\
&\quad k = 1, \ldots, T \qquad (6.3) \\
\|\boldsymbol{p}_i(t+k|t) - \boldsymbol{p}_{i-1}(t+k|t)\| &\leq \min(d_{relay,i}, d_{relay,i-1}), \\
&\quad k = 1, \ldots, T \qquad (6.4) \\
\boldsymbol{v}_i(t+T|t) &= \boldsymbol{0} \qquad (6.5) \\
\boldsymbol{p}_0(t+k|t) &= \boldsymbol{p}_0(0),\ k = 0, \ldots, T \qquad (6.6)
\end{aligned}$$

where the last equality indicates that the ground station does not move. Constraint (6.5) is a terminal hover constraint ensuring that the optimization problem at the next iteration is feasible. Indeed, applying the terminal feasible invariant set principle from Chapter 4, the remaining segments $k = 2, \ldots, T$ of the trajectories produced at time step t and combined with an extra time step in the respective hover positions $\boldsymbol{p}_i(t+T|t)$ form a feasible solution to the new optimization problem at time $t+1$. Furthermore, to prevent the actual continuous trajectories and lines of sight from cutting corners of obstacles, the obstacle sets $\mathcal{O}_{a,i}$ and $\mathcal{O}_{v,i}$ are again the actual obstacles enlarged with an appropriate safety envelope.

6.2.3 Connectivity Constraints

Compared to the basic formulation (2.3)-(2.9) of Chapter 2, the only additions are the connectivity constraints (6.3) and the maximum separation bounds (6.4). To retain a MILP formulation, the former can be expressed by substituting the single vehicle coordinates $\boldsymbol{p}_i(t+k|t) \equiv [x_i(t+k|t)\, y_i(t+k|t)\, z_i(t+k|t)]'$ in the obstacle avoidance conditions (2.28)-(2.32) by sample points along the line between subsequent agents $i-1$ and i ($i = 1, \ldots, L$):

$$\boldsymbol{p}_{i-1}(t+k|t) + \frac{l}{L_v} \left(\boldsymbol{p}_i(t+k|t) - \boldsymbol{p}_{i-1}(t+k|t)\right),\ l = 1, \ldots, L_v \tag{6.7}$$

Here L_v is the number of interpolation points. Again, due to the discrete nature of this sampling operation and the fact that visibility between the vehicles should not be lost in between two time steps, the obstacles should be enlarged with a connectivity safety envelope.

The number of sample points should be determined in function of how far the vehicles can be separated and how large a safety envelope can be tolerated.

Because they are convex, the maximum separation constraints (6.4) are easier to formulate and do not require the use of binaries. For simplicity, instead of approximating the 2-norm of the separation distance between vehicles i and $i-1$ ($i = 1, \ldots, L$) by a collection of tangent planes, we use an outer 1-norm formulation. Constraints (6.4) then become:

$$\begin{array}{lrcrcl} \forall k = 1, \ldots, T: & x_i(t+k|t) & - & x_{i-1}(t+k|t) & \leq & \min(d_{relay,i}, d_{relay,i-1}) \\ & x_{i-1}(t+k|t) & - & x_i(t+k|t) & \leq & \min(d_{relay,i}, d_{relay,i-1}) \\ & y_i(t+k|t) & - & y_{i-1}(t+k|t) & \leq & \min(d_{relay,i}, d_{relay,i-1}) \\ & y_{i-1}(t+k|t) & - & y_i(t+k|t) & \leq & \min(d_{relay,i}, d_{relay,i-1}) \\ & z_i(t+k|t) & - & z_{i-1}(t+k|t) & \leq & \min(d_{relay,i}, d_{relay,i-1}) \\ & z_{i-1}(t+k|t) & - & z_i(t+k|t) & \leq & \min(d_{relay,i}, d_{relay,i-1}) \end{array} \tag{6.8}$$

6.3 Implementation

6.3.1 Helicopter Test-Bed

Using the helicopter model from Chapter 2, the centralized MILP-based receding horizon algorithm described above was implemented on a testbed consisting of two autonomous X-Cell miniature helicopters, a ground station and a trajectory planning laptop. The X-Cell helicopters were originally developed at MIT [142, 39] and further improved and commercialized by Nascent Technology Corporation (NTC). NTC also extended the original ground station to support multi-vehicle operations and developed a simulation platform.

Figure 6-1: MIT's autonomous X-Cell helicopter, equipped with avionics box.

Figure 6-1 shows one of the autonomous X-Cells equipped with its avionics box. Power is supplied through a 50W alternator that runs off the gas engine and also charges a smart Li-Ion battery that is used as a backup power supply during interrupts. With a full tank, the endurance of the helicopter is 40 minutes. The avionics box contains an Aaeon PC-104 computer which runs the QNX real-time operating system. Among other functions, the on-board software features 1) a Kalman filter that combines GPS, IMU, and barometer

measurements to estimate the inertial position and velocity, 2) stabilizing control algorithms, 3) a waypoint follower, and 4) health management functions. Waypoints can be uploaded from the Windows-based ground station in real-time, and, through file sharing, the downloaded vehicle states are available to the ground-based trajectory planning laptop.

The communication between the ground station and the helicopters is done through Microhard Spectra NT920 wireless modems. They operate in the 900 MHz band with a 5 miles range, achieve data rates of 276 Kbps, have built-in support for Ethernet, and can be configured for both "point to multi-point" and relay operation. The latter mode is critical for beyond-line of sight operations, which is the problem of interest in this chapter. A "heartbeat"-type system monitors the communication link and keeps both the ground station and the flight computer apprised of its state. If communication is lost, a protocol is set up to re-establish it and continue normal operation, rather than to hang up or otherwise crash. If communication is lost for longer than a pre-established time out period, however, the helicopter aborts its current mission and automatically returns to the take-off point. This is one of the many autonomy and control augmentation features, which among others, include automatic take-off and landing, transition to hover in case of loss of GPS, and waypoint following or easy steering by joystick through the ground station laptop.

6.3.2 Mission Scenario

A demonstration mission scenario was developed that is based on military and civilian law enforcement requirements for beyond-line of sight urban intelligence, surveillance, and reconnaissance (ISR) operations. The target mission starts with the two helicopters stowed in the back of a ground vehicle (*e.g.*, a truck or SUV). The ground vehicle arrives at an urban site near an area of interest, the vehicles are unloaded, started, and taken off. This happens sequentially because a single operator guides the take-off process. Since the control system on the helicopter makes take-off and landing easy, (s)he need not be especially skilled.

The first (mission) helicopter is pull-started, takes off, and is put into hover mode. Next, the second (relay) helicopter is started, takes off, and is put into hover nearby. Both helicopters are under control of the ground station at all times. For safety purposes, below a certain "clearance plane", they are directly controlled by the operator through joystick steering. As soon as the vehicles are high enough, they can be guided by point-and-click operations on the ground station. Once both helicopters are above the clearance plane and in hover mode, the operator selects a location on the map at which (s)he wants video information. Next, the online trajectory planning software is started and the first series of waypoints uploaded to the relay and mission helicopter. The latter then starts flying towards the location of interest, thereby avoiding known obstacles and other constraints, while the relay helicopter adjusts its position to maintain line of sight with both the ground station and the mission helicopter.

An office park in Burlington, MA (where NTC is located), was chosen as the specific test side. Figure 6-2 shows a satellite image of the area, which is about 0.5 km by 0.3 km. The take-off area and ground station are located within the rectangle, the circle represents the target point for the mission helicopter. There are three large office buildings surrounded by parking lots and wooded areas, next to which are other buildings. A major highway (US 95) runs north-south on the left. The office buildings are about 50 m tall, the trees around 40 m. Using a combination of GPS measurements and satellite image data from Google Earth [45], a Matlab model of the relevant environment was built which is shown in Figure 6-3. The office buildings were considered separate obstacles, whereas the thick

Figure 6-2: Satellite image of the test region. The rectangle represents the take-off area and ground station, the circle is the target location. Source: Google Earth

copses of trees were treated as one large no-fly zone. Unless the mission helicopter is at an altitude that is too high for the type of ISR that we envision, flying to the target area clearly involves obscuration of the line of sight between the ground station and the mission helicopter. Although the MILP approach is flexible and could readily tackle more complicated scenarios, this specific one was chosen as a trade-off between the logistics and safety issues associated with flying two autonomous helicopters through a real obstacle environment, and sufficient complexity for a proof-of-concept demonstration of online MILP for obstacle avoidance, collision avoidance and connectivity maintenance.

6.3.3 Trajectory Planning Software

The MILP-based online trajectory optimization module was implemented using a combination of commercially available software products. The planning parameters, such as the current state and obstacle information, were processed in Matlab, the optimization model was implemented in AMPL [34], and CPLEX 9.0 [56] was used as the MILP solver. The module ran on a separate Pentium 4 laptop with 2.2 GHz clock speed, which used standard Windows file sharing with the ground station to obtain state updates from both helicopters (downloaded every second) and to upload waypoints. It was executed every 5 seconds, resulting in a new waypoint list being uploaded at 5 s intervals. We thereby made use of CPLEX's computation time limit, which was set to 4.7 s. In case no optimal solution was found within that time, the best feasible solution was returned. If no feasible solution was found in time, e.g., because of a structural infeasibility such as having been blown into an obstacle's safety envelope, the previous waypoint list was kept. Since each waypoint plan of the receding horizon strategy terminates in a safe hover state, a sequence of such

infeasibilities would not jeopardize the helicopters.

In our experiments, however, we did not encounter sequences of more than two infeasibilities. First, since most of them are related to disturbances in the environment affecting the initial state, most infeasibilities can be avoided by removing constraints on the initial time step. Second, in a preprocessing step executed at each iteration, we checked for infeasibilities such as being inside an obstacle envelope or flying faster than the maximum allowed speed and "manually" compensated for them by pushing the initial state estimate slightly into the feasible region. Provided that the vehicle has more thrust available than accounted for in the optimization problem, it can compensate for this artificial perturbance when tracking the new plan.

Another preprocessing task was to perform the appropriate coordinate transformation. Position information from the helicopters was received in GPS latitude, longitude and altitude coordinates, whereas the trajectory planning problem was solved in a local Cartesian north-east-up coordinate frame relative to the position of the ground station. Likewise, the waypoints resulting from the MILP optimization had to be transformed back before being uploaded. Since the vertices of the obstacles were stored in GPS coordinates too, they had to be transformed as well. Although for the demonstration the obstacle coordinates were stored ahead of time, new obstacles that came within reach of the planning horizon were processed (i.e., approximated by surrounding polyhedrons) at each receding horizon iteration. Overall, the complete preprocessing time to setup the updated MILP problem at each iteration took about 0.2 s.

6.4 Results

6.4.1 Planning Parameters

Before flight-testing the Burlington scenario described in Section 6.3.2, we performed Matlab and high-fidelity hardware-in-the-loop (HIL) simulations. In the hardware tests, two identical X-Cell machines or HIL simulators were employed. The parameters used in the trajectory optimization problem are given in Table 6.1. The receding horizon contained $T = 6$ time steps of $\Delta t = 2.5$ s each, corresponding to an actual planning horizon of 15 s. With a maximum velocity of 2 m/s, the 6 waypoints were thus separated by at most 5 m. Since the X-Cell's waypoint controller provided better tracking results with a 10 m resolution, however, only the 2nd, 4th and 6th waypoint were uploaded. Furthermore, for safety reasons, a minimum and maximum altitude of respectively $z_{min} = 25$ m and $z_{max} = 100$ m were enforced. The former ensured avoidance of light poles on the terrain and would give the safety pilot enough time to take over in case something went wrong.

T	6	d_{relay}	400 m	z_{max}	100 m
Δt	2.5 s	d_{safe}	10 m	$\dot{z}_{min}$	-2 m/s
N	16	v_{max}	2 m/s	$\dot{z}_{max}$	2 m/s
L_a	4	a_{max}	2 m/s^2	$\ddot{z}_{min}$	-2 m/s^2
L_v	10	z_{min}	25 m	$\ddot{z}_{max}$	2 m/s^2

Table 6.1: Parameters used in the MILP trajectory optimization

The test scenario used the following heuristic cost function:

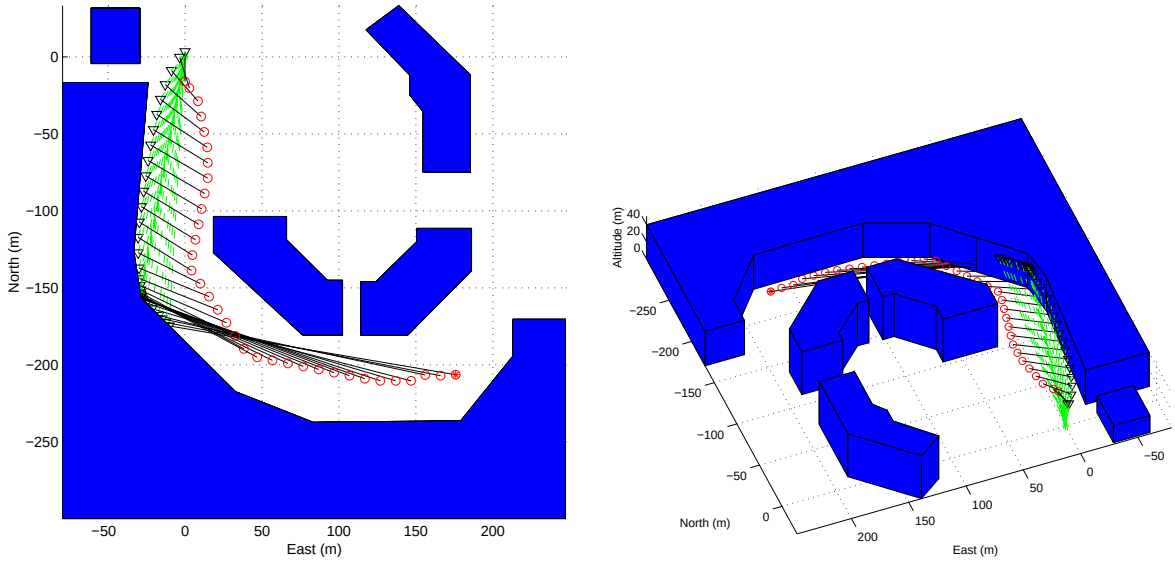
$$\begin{aligned} J_T &= \sum_{k=1}^{T} 10^{-2}|z_L(k) - z_f| + \sum_{k=0}^{T-1} 10^{-4}|\boldsymbol{a}_L(k)| + |\boldsymbol{p}_L(T) - \boldsymbol{p}_f| \\ &+ \sum_{k=1}^{T} \left(10^{-3}|\boldsymbol{v}_R(k)| + 10^{-2}|z_R(k) - z_{min}|\right) + \sum_{k=0}^{T-1} 10^{-3}|\boldsymbol{a}_R(k)| \end{aligned} \tag{6.9}$$

in which L and R respectively indicate the mission (lead) and relay helicopter, and $|\cdot|$ represents the 1-norm. This cost function seeks to guide the mission helicopter to the goal $\boldsymbol{p}_f$ by placing a high weight on the position error $|\boldsymbol{p}_L(T) - \boldsymbol{p}_f|$, while minimizing the effort of the relay helicopter to maintain line of sight. The actions of the latter are expressed in terms of its velocity and acceleration sequence. To prevent the vehicles from flying over the buildings, the cost function also penalizes altitude, as expressed by the $|z_L(k) - z_f|$ and $|z_R(k) - z_{min}|$ terms. Note that for reasons of simplicity this objective function does not account for knowledge of the environment. For a more sophisticated function with a cost-to-go term based on a visibility graph, we refer the reader to [7].

6.4.2 Simulation and Flight Test Results

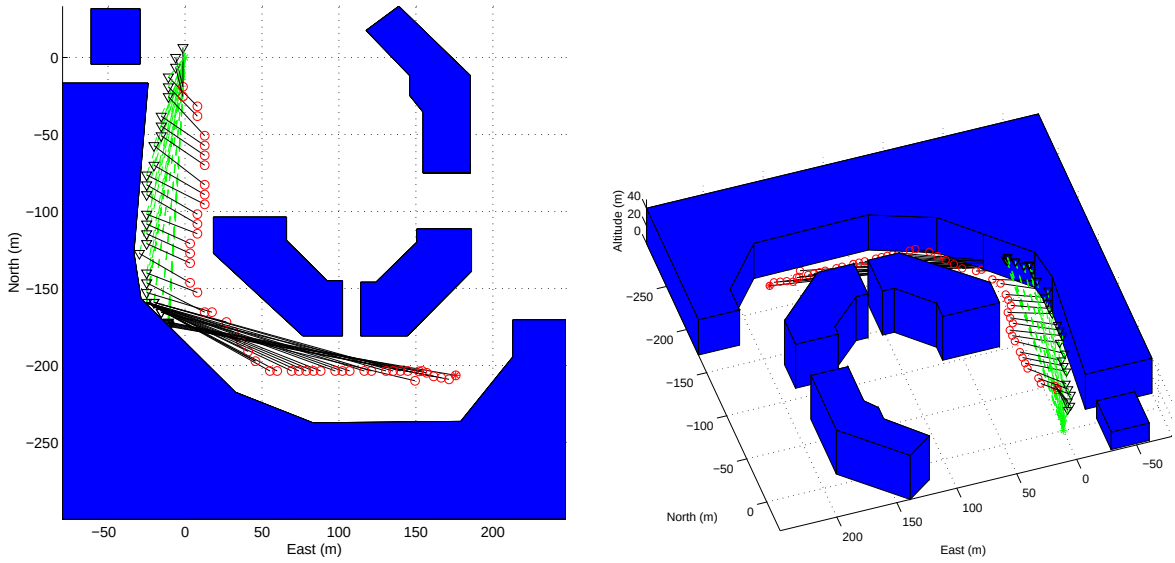
Figure 6-3 shows a Matlab simulation of the scenario in the nominal case, i.e., without disturbances or model uncertainties. The ground station is located in the origin of the local coordinate frame whose x-, y-, and z-axes are aligned with the east, north, and up directions respectively. The circles indicate the position of the mission helicopter at every 5 s time step; the triangles represent the relay agent. The initial states are hover in respectively $(0, -16, 25)$m for the mission helicopter and $(0, 3, 25)$m for the relay one. The goal, shown as a star, is located at $(176, -206, 25)$m and is reached after 33 receding horizon iterations or 165 s. The line of sight between the mission and relay helicopters at each time step is plotted in black, the connection between the ground station and the relay in green dashed line. The result shows that as the mission helicopter flies around the buildings to reach its goal, the relay moves south to maintain visibility with both the mission helicopter and the ground station. Connectivity between all agents is thus guaranteed at all times. It is also worth mentioning that at every iteration an optimal solution was always found within the time limit of 4.7 s.

Figure 6-4 shows the same scenario tested using high-fidelity HIL-simulators for both helicopters. It shows the position of the vehicles at a 5 s interval. In the HIL-sim test, the MILP algorithm operated on the same platform and with the same data that it would see in an actual flight test. The trajectory planning laptop communicated with the ground station software just as it would in the actual flight test, using GPS coordinates that were geographically correct. The ground station, in turn, communicated with the two flight computers in the same way as in the flight test, and the latter executed the same software as they would in flight. The flight computers interacted with sensors and actuators using the same low-level code as used on-board. Furthermore, the actuator commands were delivered to actual flight hardware consisting of a servoboard and servos. Only at this point did the simulation deviate from reality: servo positions were transduced by potentiometers, which were read by an A/D converter driving a high-fidelity software simulation of the vehicle flight dynamics and kinematics. Sensor outputs were also simulated, and serial outputs identical to those generated by the actual sensors were created to drive the sensor inputs of



(a) East-North projection (b) 3D plot

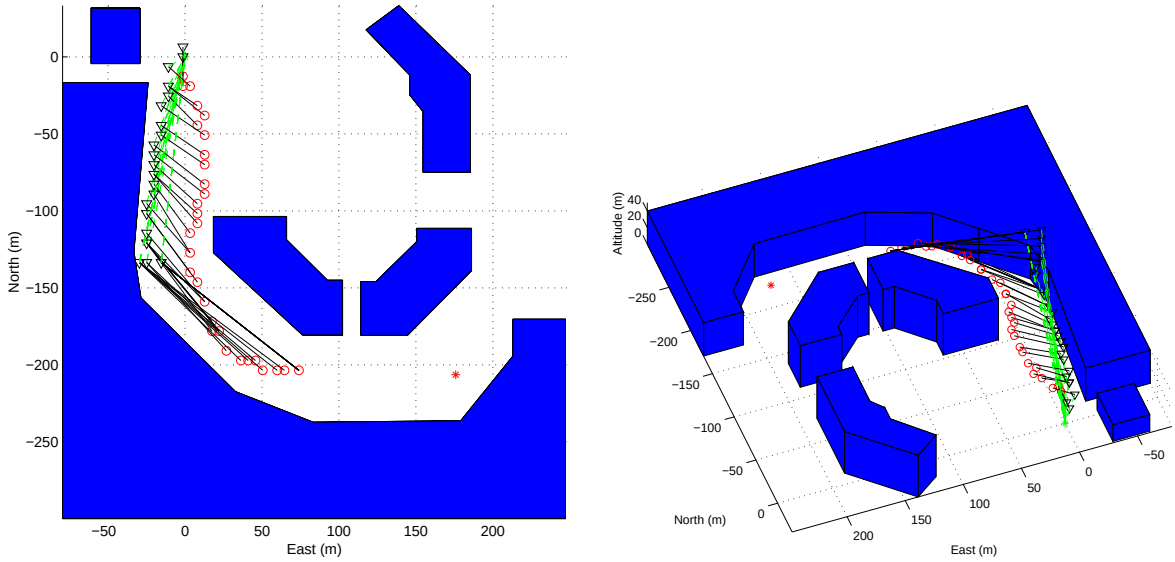
Figure 6-3: Nominal trajectories of the Burlington scenario computed in Matlab using receding horizon planning. The circles represent the mission helicopter, the triangles show the relay vehicle. The obstacles in the center are buildings; the large no-fly zone at the Western and Southern edge represents forests (enlarged with a safety boundary).



(a) East-North projection (b) 3D plot

Figure 6-4: Hardware-in-the-loop simulations of the trajectories computed online in real-time. The circles represent the mission helicopter; the triangles show the relay vehicle.

the flight computer. As can be seen from Figure 6-4, the HIL-simulation took 51 receding horizon iterations or 255 s to arrive at the goal instead of 165 s. This was mainly due to



(a) East-North projection

(b) 3D plot

Figure 6-5: Actual flight test data of the trajectories computed online in real-time. The circles represent the mission helicopter; the triangles show the relay vehicle.

the vehicle flying slower than commanded in the straight edge towards the end.

Figure 6-5 shows the result of an actual flight test. The online MILP-based trajectory planning software was initiated after the helicopters were manually taken off and brought to their starting positions by the waypoint controller. Naturally, the main difference between the HIL-simulation and the flight is the use of actual helicopters in the real environment. This adds the stresses of hardware operating in noisy environments with vibration and heat playing a role in the equipment functionality. The other major difference is the wireless communication, which is nearly perfect in the HIL-sim environment. For the flight test, an actual relay protocol was implemented using the Microhard 900 MHz Ethernet modems described in Section 6.3. These modems performed very well, maintaining communication, as planned, when the line of sight was blocked. As seen in Figure 6-5, visibility between the ground station and the mission helicopter was indeed lost during the flight. An indirect communication link, however, was maintained through the relay helicopter. Based on this setup, we know that all waypoints executed by the mission helicopter were sent to it by way of the relay, and that all waypoints executed by the latter were uplinked directly.

Finally, the flight differed from the HIL-simulation because real sensors were used in the actual mission. In particular, this meant that GPS coordinates were based on actual measurements, which sometimes suffer dropouts and loss of accuracy, especially in urban settings. Our scenario environment can be considered suburban, so for the most part GPS performance was good. Other sensor measurements on-board the helicopter are reliable and accurate, and performed flawlessly during the mission.

The first part of the flight looks very similar to the HIL-sim results. About 65% through the mission, however, a faulty power supply on-board the relay helicopter caused the communication with the ground station to be temporarily interrupted. As such, both helicopters ran out of waypoint updates and safely hovered at the last waypoint of their latest plan (approximately $(-25, -120, 25)$ for the relay and $(22, -178, 25)$ for the mission helicopter).

As a safety procedure, the relay helicopter then entered a "safe return home" mode, which consists of first climbing to an altitude of 100 m and then returning to the position from where it started. This climb is visible in the last few data points shown in the 3D plot of Figure 6-5. Once in safe return home mode, the relay helicopter ignored the additional waypoints that were uploaded to it. However, communication picked up again just after entering this mode and right before the mission helicopter would have initiated it. As a result, after having hovered around location $(22, -178, 25)$, the mission helicopter continued receiving and executing the MILP updates. It thereby maintained line of sight with the climbing relay helicopter until the mission was manually aborted. This in fact highlighted the flexibility and adaptive replanning properties of the receding horizon MILP approach. Furthermore, this was the first time that the X-Cell helicopters were guided by an on-line MILP-based trajectory planner around actual buildings, thus providing a successful proof-of-concept of the use of MILP for in-air obstacle and collision avoidance.

6.5 Distributed Planning Strategy

Because the centralized formulation presented in Section 6.2 accounts for the states and inputs of all vehicles as unknowns in the optimization problem, its computation time scales exponentially with the number of agents. To reduce the computational complexity, it is therefore more sensible to use a distributed strategy in which each helicopter only computes its own trajectory. One possible approach is that all vehicles compute their trajectories according to a cyclic order corresponding to the relay sequence and thereby always account for the latest plans of the other vehicles. For vehicles j' that already updated their trajectory in the current cycle, the latest plan is the newly optimized trajectory. For vehicles j'' that follow later in the cycle, the latest plan is the one computed during the previous iteration. The current helicopter j can then consider the plans of all other vehicles j' and j'' as fixed waypoints $\hat{\boldsymbol{p}}_{j'}(\cdot)$ and $\hat{\boldsymbol{p}}_{j''}(\cdot)$, such that they enter as constraints in the optimization problem and not as unknowns. As in the centralized formulation, feasibility at all times can be guaranteed by constraining the intermediate trajectories of all helicopters to terminate in hover states at distinct locations.

6.5.1 Distributed Cooperative Algorithm

Starting with the leader L who optimizes its mission-dependent cost function, a possible objective function for each subsequent relay agent is to minimize the distance to the previous vehicle in the cycle (or thus to the next one in the relay order). Assuming an initial feasible relay configuration at $t = 0$, the preceding strategy is formalized by the following algorithm, executed at each time step t:

Start: Start at time t.

Step 1: Let leader L perform the following actions:

Step 1a: Solve the following optimization problem:

$$J_L^* = \min \sum_{k=0}^{T-1} \ell_{L,k}(\boldsymbol{x}_L(t+k|t), \boldsymbol{u}_L(t+k|t), \boldsymbol{x}_f) + f_L(\boldsymbol{x}_L(t+T|t), \boldsymbol{x}_f) \tag{6.10}$$

subject to:

$$
\begin{aligned}
\boldsymbol{x}_L(t+k+1|t) &= \boldsymbol{A}_L\boldsymbol{x}_L(t+k|t) + \boldsymbol{B}_L\boldsymbol{u}_L(t+k|t), \\
&\quad k = 0, \ldots, T-1 \\
\boldsymbol{x}_L(t|t) &= \hat{\boldsymbol{x}}_L(t|t-1) \\
\boldsymbol{x}_L(t+k|t) &\in \mathcal{X}_L(k),\ k = 1, \ldots, T \\
\boldsymbol{u}_L(t+k|t) &\in \mathcal{U}_L(k),\ k = 0, \ldots, T-1 \\
\boldsymbol{p}_L(t+k|t) &\notin \mathcal{O}_{a,L}(t),\ k = 1, \ldots, T \\
\boldsymbol{p}_L(t+k|t) + \lambda\left(\hat{\boldsymbol{p}}_{L-1}(t+k|t-1) - \boldsymbol{p}_L(t+k|t)\right) &\notin \mathcal{O}_{v,L}(t) \cap \mathcal{O}_{v,L-1}(t-1), \\
&\quad 0 \leq \lambda \leq 1,\ k = 1, \ldots, T \\
\|\boldsymbol{p}_L(t+k|t) - \hat{\boldsymbol{p}}_{L-1}(t+k|t-1)\| &\leq \min(d_{relay,L}, d_{relay,L-1}), \\
&\quad k = 1, \ldots, T \\
\|\boldsymbol{p}_L(t+k|t) - \hat{\boldsymbol{p}}_j(t+k|t-1)\| &\geq d_{safe},\ j = 0 \ldots L-1 \\
\hat{\boldsymbol{p}}_j(t+T|t-1) &= \hat{\boldsymbol{p}}_j(t-1+T|t-1), \\
&\quad j = 0, \ldots, L-1 \qquad (6.11) \\
\boldsymbol{v}_L(t+T|t) &= \boldsymbol{0} \qquad (6.12)
\end{aligned}
$$

Step 1b: Send new waypoints $\hat{\boldsymbol{p}}_L(t+k|t)$ to all other agents $j = 1, \ldots, L-1$ through the relay network.

Step 2: Let all relay agents $j = L-1, L-2, \ldots, 1$ *subsequently* perform the following actions:

Step 2a: Solve the following optimization problem:

$$
J_j^* = \min \sum_{k=1}^{T} q|\boldsymbol{p}_j(t+k|t) - \hat{\boldsymbol{p}}_{j+1}(t+k|t)| + \sum_{k=0}^{T-1} r|\boldsymbol{u}_j(t+k|t)| \qquad (6.13)
$$

subject to:

$$
\begin{aligned}
\boldsymbol{x}_j(t+k+1|t) &= \boldsymbol{A}_j\boldsymbol{x}_j(t+k|t) + \boldsymbol{B}_j\boldsymbol{u}_j(t+k|t), \\
&\quad k = 0, \ldots, T-1 \\
\boldsymbol{x}_j(t|t) &= \hat{\boldsymbol{x}}_j(t|t-1) \\
\boldsymbol{x}_j(t+k|t) &\in \mathcal{X}_j(k),\ k = 1, \ldots, T \\
\boldsymbol{u}_j(t+k|t) &\in \mathcal{U}_j(k),\ k = 0, \ldots, T-1 \\
\boldsymbol{p}_j(t+k|t) &\notin \mathcal{O}_{a,j}(t),\ k = 1, \ldots, T \\
\|\boldsymbol{p}_j(t+k|t) - \hat{\boldsymbol{p}}_{j+1}(t+k|t)\| &\leq \min(d_{relay,j}, d_{relay,j+1}), \\
&\quad k = 1, \ldots, T \qquad (6.14) \\
\boldsymbol{p}_j(t+k|t) + \lambda\left(\hat{\boldsymbol{p}}_{j+1}(t+k|t) - \boldsymbol{p}_j(t+k|t)\right) &\notin \mathcal{O}_{v,j}(t) \cap \mathcal{O}_{v,j+1}(t), \\
&\quad 0 \leq \lambda \leq 1,\ k = 1, \ldots, T \qquad (6.15) \\
\|\boldsymbol{p}_j(t+k|t) - \hat{\boldsymbol{p}}_{j-1}(t+k|t-1)\| &\leq \min(d_{relay,j}, d_{relay,j-1}), \\
&\quad k = 1, \ldots, T \qquad (6.16) \\
\boldsymbol{p}_j(t+k|t) + \lambda\left(\hat{\boldsymbol{p}}_{j-1}(t+k|t-1) - \boldsymbol{p}_j(t+k|t)\right) &\notin \mathcal{O}_{v,j}(t) \cap \mathcal{O}_{v,j-1}(t-1), \\
&\quad 0 \leq \lambda \leq 1,\ k = 1, \ldots, T \qquad (6.17)
\end{aligned}
$$

$$\|\boldsymbol{p}_j(t+k|t) - \hat{\boldsymbol{p}}_{j'}(t+k|t)\| \geq d_{safe},\ j' = j+1,\dots,L \tag{6.18}$$
$$\|\boldsymbol{p}_j(t+k|t) - \hat{\boldsymbol{p}}_{j''}(t+k|t-1)\| \geq d_{safe},\ j'' = 0,\dots,j-1 \tag{6.19}$$

$$\hat{\boldsymbol{p}}_{j''}(t+T|t-1) = \hat{\boldsymbol{p}}_{j''}(t-1+T|t-1),\quad j'' = 0,\dots,j-1 \tag{6.20}$$
$$\boldsymbol{v}_j(t+T|t) = \boldsymbol{0} \tag{6.21}$$

Step 2b: Send new waypoints $\hat{\boldsymbol{p}}_j(t+k|t)$ to all other agents $j'' = 1,\dots,j-1$ and $j' = j+1,\dots,L$ through the relay network.

End: End before time $t+1$.

Notice that the cost function (6.10) for the lead vehicle L does not account for the other vehicles and only optimizes a specific mission objective. The cost functions (6.13) for the subsequent relay agents j, however, seek to minimize the 1-norm distance to the previous agent $j+1$ in the decreasing planning sequence, i.e., to the vehicle that just updated and broadcasted its new waypoints $\hat{\boldsymbol{p}}_{j+1}(t+k|t)$. Constraints (6.14)-(6.15) express the connectivity requirement with this previous agent, whereas constraints (6.16)-(6.17) ensure connectivity with the next agent $j-1$ in the update sequence, using the latter's previous waypoints $\hat{\boldsymbol{p}}_{j-1}(t+k|t-1)$ with an additional time step in the terminal hover location given by equation (6.20). Finally, the collision avoidance constraints (6.18) and (6.19) also account for the latest waypoint plans of all vehicles in the cycle, namely $\hat{\boldsymbol{p}}_{j'}(t+k|t)$ for agents $j' = j+1,\dots,L$ and $\hat{\boldsymbol{p}}_{j''}(t+k|t-1)$ for agents $j'' = 0,\dots,j-1$.

The planning sequence for all agents $j = L, L-1,\dots,1$ should be completed before the next time step. To ensure this, each vehicle in the sequence should be allocated a time slot in which to compute its trajectory update. An *a priori* feasible solution will always be available, consisting of the remainder of the previous plan with the additional time step in the terminal hover state. Therefore, if the vehicle cannot find a better solution within the allotted time slot, it can always resort to this backup plan. Constraints (6.11) and (6.20) determine the last time step of this *a priori* solution for the vehicles following later in the update cycle. As such, together with the terminal hover constraints (6.12) and (6.21), feasibility of the planning problem for all subsequent agents and time steps is guaranteed. Because it is similar in nature to the feasibility and safety proofs from Chapters 4 and 5, a formal proof of this statement is omitted. It would again consist of the explicit construction of the feasible solution just described.

6.5.2 Results

Simulation results of the preceding distributed algorithm are shown in Figures 6-6 and 6-7. The former revisits the Burlington scenario with an extra obstacle placed in the field. To maintain connectivity with the ground station a second relay vehicle must be introduced which, starting from $(0, 16, 25)$m, positions itself near the lower right corner of the new obstacle. On average, the total cycle for the three vehicles at each receding horizon iteration took 1.3 s, indicating that the computational complexity of the distributed strategy scales much better with the number of relay agents than the centralized approach.

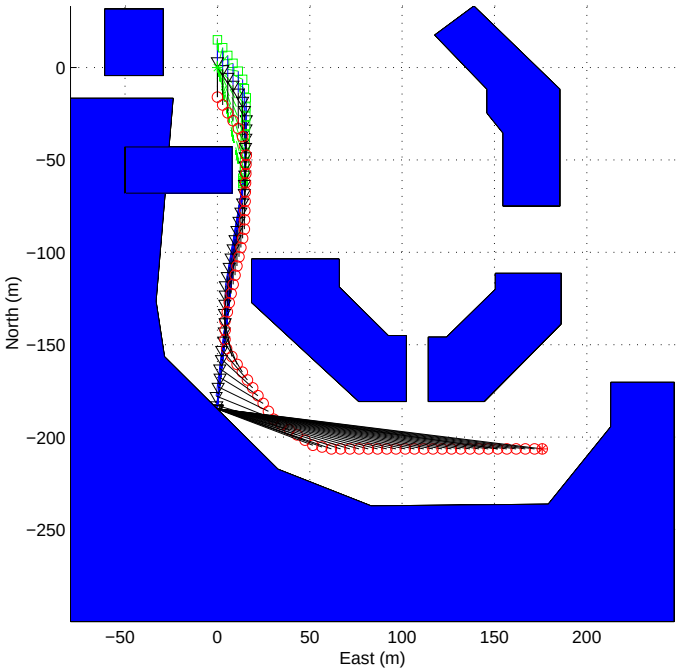


Figure 6-6: Simulation result for distributed Burlington scenario with extra obstacle and second relay vehicle.

In general, for mission purposes, the lead vehicle might be constrained to stay at a lower altitude, whereas the relay agents might be able to position themselves anywhere in the environment. A scenario in which the relay helicopter takes advantage of altitude changes to maintain connectivity is plotted in Figure 6-7. This particular environment represents a part of the MIT campus. Both helicopters again start close to the origin where the ground station is located. The mission helicopter must now maneuver to a hover state near $(-100, 80, 10)$m while staying low over the building. The altitude of the relay helicopter is unconstrained and, as such, it climbs to a higher altitude that is sufficient to maintain line of sight with both the mission helicopter and the ground station. This again highlights the flexibility of the MILP approach. Using the same planning parameters as in the Burlington scenario, each two-vehicle cycle in this scenario took 1.0 s on average.

6.6 Conclusion

This chapter presented a centralized and distributed receding horizon strategy for online connectivity-constrained MILP-based trajectory planning of autonomous vehicles through cluttered environments. The centralized approach was successfully implemented on two X-Cell helicopters and tested in a real-world scenario. The test-bed, mission and first time proof-of-concept results of online multi-helicopter operation in an actual obstacle environment were described in detail. The results show that MILP is a flexible framework that can effectively handle obstacle avoidance, collision avoidance, and connectivity constraints in real-time. Simulations of a distributed planning strategy indicated that the approach is scalable to multi-relay networks.

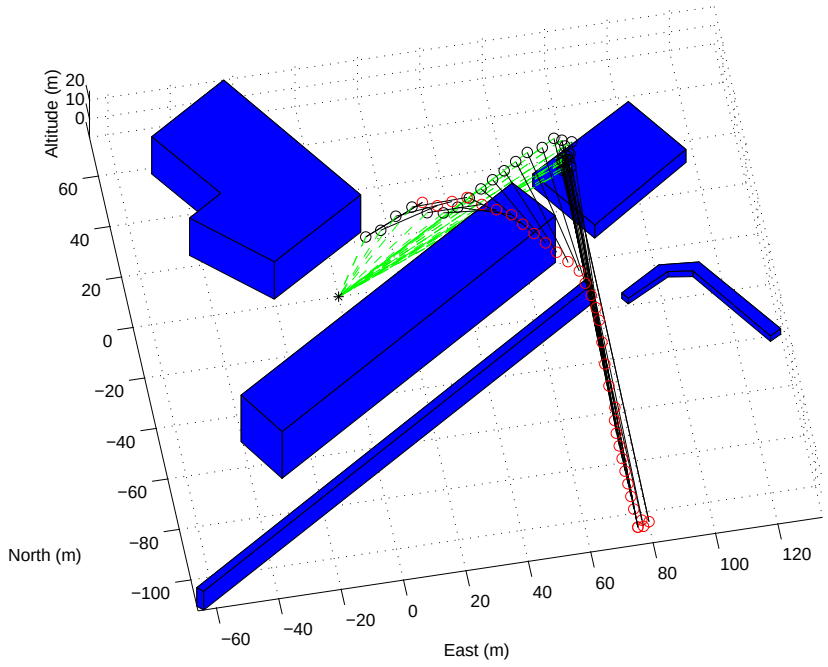


Figure 6-7: Simulation result for distributed scenario in which the mission helicopter must fly over the building (MIT's Simmons Hall) and the relay has to gain altitude.

Chapter 7

Implementation of MILP-based UAV Guidance for Human/UAV Team

This chapter discusses the implementation of a MILP-based guidance system as part of a manned vehicle – UAV mission system. The overall system enables an operator in a manned aircraft to issue mission level commands to an autonomous aircraft in real-time. A natural language interface allows the manned and unmanned vehicles to communicate in languages understood by both agents. A task scheduler transforms the commands into a dynamic mission plan consisting of task waypoints. These are then given to the MILP-based trajectory optimizer, which safely guides the vehicle through a partially-known environment in real-time. Integrated simulation and flight-test results are presented that used an F-15 and an autonomous T-33 equipped with Boeing's UCAV avionics package. These activities were part of the Capstone Demonstration of the DARPA-sponsored Software Enabled Control effort. The flight-tests mark the first time that an onboard MILP-based guidance system was used to control a UAV. They also mark the first time that a natural language interface was used by a manned vehicle to task a UAV in real-time.

7.1 Introduction

Recent advances in guidance and autonomy technology have enabled some UAVs to execute simple mission tasks without human interaction. Most of these tasks are pre-planned using reconnaissance or environment information. For example, air operations are executed according to an Air Tasking Order (ATO), which may take up to 72 hours to plan, task and execute [153]. In volatile situations, however, information about the vehicle's operating environment may be limited: a detailed map of the environment might not be available ahead of time, and obstacles might be detected while a mission is carried out. In such situations, task planning flexibility and safe trajectory solutions are essential to the survivability and success of the autonomous system: the vehicle's guidance and mission planning systems must possess enough intelligence (and processing power) to recognize and react to changes in the operating conditions.

The complexity of the above problem increases when more than one agent is introduced. For example, if other autonomous agents are added to the mission scenario, then all vehicles must resolve information regarding the impending actions of the other vehicles. Similarly,

if a manned agent is introduced, the autonomous vehicles must also possess the capability to effectively communicate and coordinate their actions with the manned vehicle. Most unmanned vehicles, however, do not exhibit this level of performance. Intelligent mission and guidance systems providing the flexibility and cooperative behavior needed to complete an entire mission autonomously are therefore a topic of active research [67, 122, 25, 47].

This chapter discusses the development, implementation and evaluation of such a system containing a manned vehicle and a UAV operating in a partially-known environment. It enables the operators of the manned aircraft to issue tasks and mission-level commands to the unmanned aircraft in real-time using a natural language interface. The latter translates English sentence commands from the crew to a set of codes understood by the UAV, and vice versa. A task scheduler then transforms these commands into input data of an online MILP-based trajectory optimization problem, using the feasible receding horizon planning strategy from Chapter 4. The overall mission system thus transforms the natural language commands of the manned aircraft operators into a mathematical programming problem producing real-time trajectories that implement the dynamic mission plan of the UAV.

A number of challenges had to be overcome during the development of this system. First, the mechanism allowing both vehicles to communicate with one another needed to be designed. Second, the UAV guidance technology had to be made robust to changes and threats in the vehicle's environment – including changing wind conditions, no-fly zones and other obstacles,– and produce safe trajectories through the partially-known environment. Third, since the system was intended for real-time missions, all developed algorithms needed to reach a solution and resolve any unexpected issues reliably in a pre-defined period of time.

As part of the DARPA-sponsored Software Enabled Control (SEC) Program, the system was implemented on a test-bed consisting of an F-15, acting as the manned aircraft, and a T-33 augmented with Boeing's UCAV avionics package, acting as the autonomous vehicle. During the SEC Capstone Demonstration in June 2004, it was successfully flight-tested at the NASA Dryden Flight Research Center. These flight-tests mark the first time that a natural language interface was used by a manned vehicle to task a UAV in real-time, and the first time that a MILP-based guidance system was used to control a UAV.

The chapter is organized as follows. Section 7.2 gives an overview of the SEC experiment and the associated technology development. Section 7.3 describes the natural language interface and Section 7.4 covers the task scheduling and communication components. The implementation of the real-time trajectory planning software is discussed in Section 7.5. Simulation and flight-test results are presented in Section 7.6.

7.2 Experiment and Technology Overview

7.2.1 Mission Scenario

As originally discussed in [94], as part of the DARPA-sponsored Software Enabled Control Program, we were was tasked with developing a mission system and flight-test scenario that exhibited UAV technology developed at MIT. For this demonstration, two flight assets were available: a Boeing F-15E fighter jet (similar to the aircraft shown in Figure 7-1(a)) and a Lockheed T-33 trainer fighter jet (similar to the one shown in Figure 7-1(b)) that was equipped with Boeing's UCAV avionics package. The former was to be flown by a pilot and will be referred to as the Fixed-Wing (FW) vehicle. The latter was to be guided by our technology and will be referred to as the Unmanned Aerial Vehicle (UAV). Besides these

(a) Boeing F-15E Strike Eagle

(b) Lockheed T-33 Shooting Star

Figure 7-1: SEC Capstone Demonstration test vehicles

aircraft, a ground station receiving state and user-defined information from both vehicles was available to monitor the experiment.

To enable a hard real-time execution, our demonstration software needed to be integrated with Boeing's Open Control Platform (OCP) [106] and loaded onto a laptop fitted in each aircraft. The OCP software provided an aircraft interface that included the following abilities: 1) send and receive state and user-defined data between both aircraft using a Link-16 communications interface; 2) receive the current vehicle state data; 3) send a set of pre-defined commands to the aircraft avionics system which include Set and Hold Turn Rate, Set and Hold Speed, Set and Hold Altitude, Set and Hold Heading; and 4) memory storage and time frame execution.

Given these demonstration resources a mission scenario (shown in Figure 7-2) was developed in which the UAV performs tasks in support of the FW vehicle:

Mission

A manned fighter aircraft (FW) and a UAV will work together on a mission to collect images of a possible site in enemy territory. The FW Weapon Systems Officer (WSO) will communicate with the UAV using a natural language interface, which allows the FW WSO to speak with the UAV using normal sentence commands. The UAV will perform the reconnaissance for the mission in a partially-known environment, and the FW WSO will decide how the UAV will be used to accomplish the mission goals. The UAV will possess the ability to detect threats and collect images, whereas, if applicable, the FW vehicle will be able to deliver weapons. Since the environment is only partially-known, there may be threats to both the manned and unmanned aircraft.

Starting Condition

The UAV will start in a pre-defined loiter pattern; the FW vehicle will be flying an air-patrol near the enemy territory. The environment is partially-known and updated in real-time to both the UAV and the FW. A pop-up threat may arise en route to the search site, which is currently unknown.

Mission Narrative

1: The FW vehicle is commanded to gather information and possibly destroy an enemy site located in unknown territory. Because of the mission risk, the FW vehicle assigns

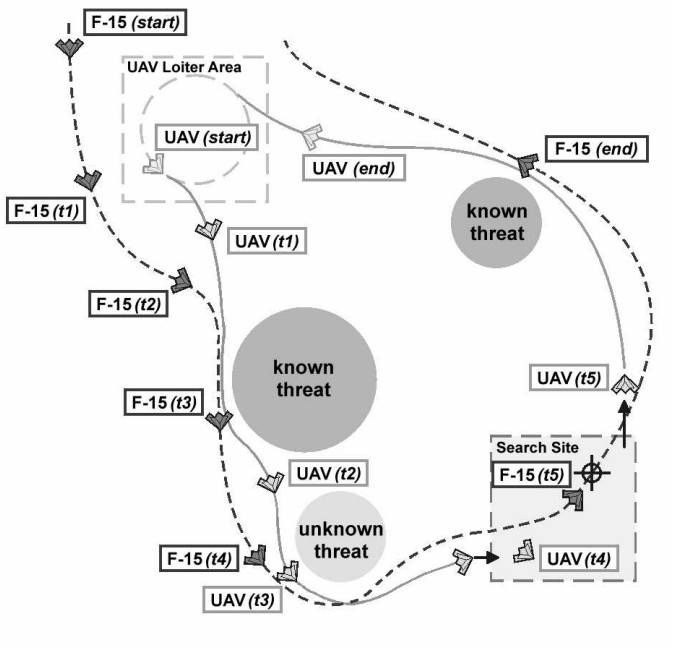


Figure 7-2: Overview of the MIT flight experiment

the UAV, stored in a nearby airspace volume, to gather information at the designated site. The UAV leaves the loiter area and moves toward the designated task area. The F-15 follows behind at a higher altitude and a safe distance.

2: The UAV is informed of a pop-up threat en route to the task area. The UAV accounts for the threat dynamically, automatically generates a revised feasible trajectory around the threat and other no-fly zones, while notifying the FW vehicle of the threat's position.

3: As the UAV moves within a few minutes of the task location, the UAV notifies the FW vehicle of its location. At this point, the FW will provide the UAV with the exact ingress and egress conditions into and out of the search area. The UAV modifies its flight path to arrive at the site as commanded.

4: The UAV enters the site, notifies the FW vehicle of its location and begins its search for the target.

5: The UAV identifies the target and sends an image to the FW vehicle for evaluation. The FW commands the UAV to return to its original loiter area, while it prosecutes the target.

Exit Conditions

The UAV safely returns to the original pre-defined loiter location; the FW vehicle returns to flying an air-patrol near the enemy territory.

7.2.2 Technology Development

The above mission scenario allowed us to demonstrate technology developments in several areas, leading to three distinct software components:

1: *Natural Language Interface* — This component interprets and converts normal sentence commands from the humans onboard the FW vehicle into data the UAV can understand and use, and vice-versa. It enables the FW WSO to give high level mission commands to the UAV in English, e.g. "Search this region for threats", rather than low level guidance commands such as "Turn left" or "Speed up". As such, the Natural Language Interface is aimed at minimizing the workload of the FW WSO when interacting with the computer-based UAV.

2: *Task Scheduling and Communications Interface* — The primary goal of this component is to interpret the command data from the Natural Language Interface and develop a series of tasks the vehicle can perform. The mission tasks that were developed included flying to a waypoint X, entering a loiter pattern, and performing a search pattern. The component also contains the communications processing module that provides the FW WSO with the authority to send task commands and receive status updates, threat and obstacle avoidance information, and acknowledgement messages.

3: *MILP-based Trajectory Generation* — After the Natural Language Interface and Task Scheduling component have converted the mission steps into a series of tasks for the vehicle to perform, the Trajectory Generation Module guides the vehicle from one task location to the next. Approximate time-optimal trajectories that account for the current state of the vehicle and the knowledge of the environment are computed online using mixed-integer linear programming. Since in the mission scenario the environment is only partially-known and explored in real-time, the MILP guidance algorithm uses the feasible receding horizon planning strategy presented in Chapter 4.

Each of these components addresses a capability required to perform the above mission. Figure 7-3 shows a block diagram representation of the integrated FW and UAV demonstration system. In the following sections, the development and integration of the three technologies is discussed. The focus, however, is placed on the Trajectory Generation Module.

7.3 Natural Language Parsing and Interfacing

The main goal of using a Natural Language Interface (NLI) for interacting with a computer-based system is to minimize the workload on the operator. Using normal English sentence commands indeed allows the FW WSO or pilot to communicate efficiently and effectively with the UAV, as if it were a human wingman. The NLI module developed for the demonstration consists of two major components. The first one takes sentence commands from the FW WSO and turns them into a coded command that is sent to the UAV over Link-16. The second component takes a coded command set from the UAV and converts it into a natural language response for the FW WSO to interpret.

A sample dialog between the FW WSO and the UAV, in which the latter is commanded to search a pre-defined region containing a potential threat, could be as follows:

FW: "UAV 7, this is Eagle 3."

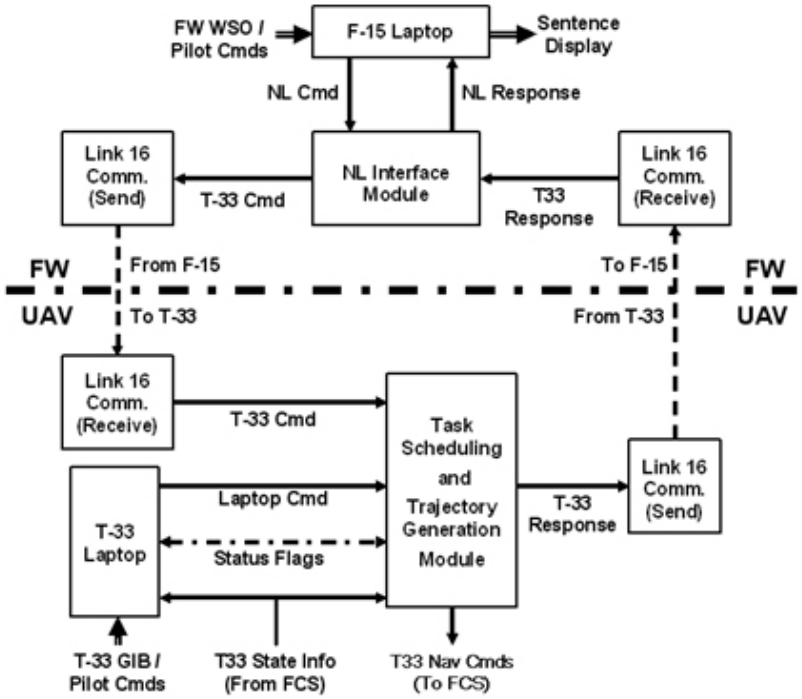


Figure 7-3: Block diagram of the MIT SEC Capstone Demonstration system. "FCS" stands for Flight Control System, "GIB" for Guy-in-Back (i.e., the rear-seat operator).

UAV: "Go ahead, Eagle 3."

FW: "Add new mission task. Proceed to location Echo-Charlie 5 in minimum time. Search this region for threats and wait for further instructions after the task is completed"

UAV: "Roger. Acknowledge task information - proceeding to location Echo-Charlie 5."

FW: "Eagle 3, out."

The NLI module analyzes the natural sentences produced by the FW WSO using parsing, which is the process of converting an input sentence, e.g., "Proceed to location Echo-Charlie 5 in minimum time," into a formal representation. The latter is typically a tree structure which can in turn be translated into an explicit formal command. In our system, parsing consists of first applying entity extraction to all the individual concepts (e.g., "Eagle 3" or "Echo-Charlie 5") and then combining these concepts through cascades of finite-state transducers using techniques derived from those described in [118].

Because the vocabulary of the SEC experiment is much smaller than for generic applications, the level of ambiguity is reduced, which makes parsing easier than on more open text. However, compared to other information processing tasks, this deployment required a particular emphasis on the safety of the parsing process. The stability of the runtime module was achieved by shifting the complexity of the system toward the off-line model compilation phase (for which there are much less stringent stability requirements). This makes the runtime process much simpler. In addition, the runtime process consists mostly of finite-state operations whose algorithm can be proven correct and for which the input

finite-state machines can be checked for particular formal properties. This approach has the additional benefit of providing a very efficient processing time, which can be bounded explicitly.

Because of budgetary constraints, our team was unable to incorporate voice recognition in the system. Instead, a set of useful commands was chosen that are available to the FW WSO through pre-defined experiment keys on the F-15 laptop. When an experiment key is pressed, the associated sentence is sent to the NLI module. It is then parsed and converted into a nine number code used by the UAV as an input command. This code uses the following protocol:

Message Description

Cmd ID: *<long integer>*

Cmd Data Words 1-8: *<double>*

The Command Identification (Cmd ID) value denotes the type of command sent between the vehicles, whereas the Command Data Words contain the actual information. For example, Cmd ID 102 may represent the "Command Acknowledge" data set, and Cmd ID 106 may represent the "New / Change Current Task" data set. The Cmd Data Words corresponding to Cmd ID 102 may then consist of a coded representation of "Proceeding to location Echo-Charlie 5". Because of user bandwidth limitations in the demonstration, each command word identifies a maximum of eight data words.

7.4 Task Scheduling and Communications Interfacing

The task scheduling and communications processing components are designed to centralize all of the UAV's mission processing in one module. Together with the Natural Language Interface, it provides flexibility for an operator to insert and change mission tasks during the operation. The UAV software keeps track of the mission tasks, waypoint locations and known obstacles to pass on to the guidance algorithm.

The communications processing component provides the FW WSO with the authority to send task commands and receive status updates, threat or obstacle avoidance information and acknowledgement messages. It also provides the ground operators monitoring the UAV during the demonstration with the ability to override the guidance system in the event of an emergency or error. The system sends threat and override information to the FW WSO before any status or update information in an effort to send the most important data relevant to the demonstration before any auxiliary information. Input/Output data are processed every 1 Hz frame before the task planner and guidance step to ensure that the most up-to-date information is used by the UAV trajectory planner.

The task scheduling component allows a user to plan a number of tasks using a pre-defined list or as programmed during a mission. Since many missions are pre-planned, the system allows an operator to initiate a pre-defined mission task or to modify or create a mission plan by entering specific task parameters. The list of mission tasks includes: Fly to Waypoint X, Loiter Pattern, Search Pattern, Classify Target, Attack Target, Battle Damage Assessment, and Return-to-Base. For each of these task options, the user must provide the ingress and egress conditions, and the size and location of the rectangular task area, given by the lower left and upper right coordinates. In addition, he or she has the option of providing (in real-time via the NLI) the optimization metric used by the trajectory

generation algorithm (i.e., minimum time, minimum fuel, or the amount of time to finish the task).

Next, the operator can either give the vehicle a new task or change the current task it is performing. A "New Task" command is added to the end of the UAV task list and is executed after all of the tasks currently in the scheduler have been completed. A "Change Task" command, on the other hand, modifies the current task performed by the UAV. Once a task is completed, it is removed from the list. After each of these actions, an acknowledgement is sent to the FW WSO and the updated task information is included in the data sent to the Trajectory Generation Module.

To reduce the complexity of the demonstration system, only the current task could be modified, although future versions of this system will have the capability to change any of the tasks in the scheduler. Furthermore, because of communication link bandwidth constraints, the FW WSO did not have the capability to define a new task or adjust parameters in the current task manually. Instead, he was able to command the vehicle to perform tasks from a pre-defined library using the experiment keys on the FW laptop.

7.5 MILP-based Trajectory Generation Module

After the Natural Language Interface and Task Scheduling components have converted the mission steps into a series of tasks for the UAV to perform, the Trajectory Generation Module guides the vehicle from one task location to the next, i.e., from an initial state to a desired one, through an obstacle field while optimizing a certain objective. For the demonstration, 2D scenarios were considered in which no-fly zones or "obstacles" are detected while the mission is carried out, but such that the environment is always fully characterized inside a certain detection region around the aircraft. The demonstration scenarios used a circular region of radius 9 mi and assumed that all obstacles $\mathcal{O}$ within that radius were static. The formulation could, however, be easily generalized to account for any detection shape, such as a radar cone, and for unknown areas within that shape. Since the environment is only partially-known and further explored in real-time, a feasible receding horizon planning strategy is implemented using left and right turning loiter circles as terminal feasible invariant sets.

7.5.1 MILP Formulation

Dynamic Model

System identification experiments using the UAV DemoSim simulation software provided by Boeing gave us insight into the velocity response of the UAV. A piecewise linear first-order approximation was deemed to be sufficient for guidance purposes. The time constant and DC gain of the transfer function were identified for a discrete set of forward velocities (from 350 fps to 500 fps with a resolution of 10 fps) and stored in a look-up table. At each iteration of the receding horizon strategy, the model corresponding to the velocity at that time step was used, thus linearizing the nonlinear response into several LTI modes scheduled around the initial velocity.

Taking the desired planar inertial velocity as input then gives the continuous-time state space model:

$$\begin{aligned} \ddot{x}(t) &= -\tfrac{1}{\tau_l}\dot{x}(t) + \tfrac{k_l}{\tau_l}\dot{x}_{cmd}(t) \\ \ddot{y}(t) &= -\tfrac{1}{\tau_l}\dot{y}(t) + \tfrac{k_l}{\tau_l}\dot{y}_{cmd}(t) \end{aligned} \tag{7.1}$$

where τ_l is the time constant and k_l is the gain corresponding to the l^{th} LTI mode. As in equation (5.3), it was discretized using the bilinear transform. Since the typical time constant of the T-33 velocity response was around 9.7 s, a time step of $\Delta t = 10$ s was used. With additional constraints on speed and acceleration, this model produced good results for tasks requiring intensive waypoint tracking and sharp turns (i.e., the loiter and search tasks). For less aggressive trajectories with more or less constant speed, such as when transitioning between two task areas, the simpler (2D) double integrator model (2.20) was used that does not distinguish between the different LTI modes.

As discussed in Chapter 2, since it takes a certain time to compute the trajectory, the initial state $\boldsymbol{x}(0)$ should be an estimate of the vehicle's state when the plan is actually implemented. In our case, the computation delay was approximately 1 s. In addition, a 1.2 s actuator delay had to be accounted for. To obtain an estimate of the initial state at the next iteration, the dynamics were thus propagated forward according to the previous plan for 2.2 s.

To ensure that the planned trajectory respected the velocity limits of the vehicle, the same maximum and minimum speed constraints were included as in inequalities (2.21) and (2.23)-(2.25). For the reference velocity model (7.1) these were formulated in terms of the velocity commands $\dot{x}_{cmd}(k)$ and $\dot{y}_{cmd}(k)$, for the double integrator model in terms of the state velocity variables $\dot{x}(k)$ and $\dot{y}(k)$. The minimum and maximum bounds were respectively set to $v_{\min} = 400$ fps and $v_{\max} = 450$ fps, using a polygonal approximation with $N = 32$. To avoid infeasible problems in the event the actual ground speed fell outside these bounds (e.g., because of wind gusts), their values were accordingly adapted online.

Since the dynamics are homogeneous in the x- and y-coordinates and as such ignore differences in the lateral and longitudinal aircraft dynamics, we added constraints capturing limits on turn rate and on forward and lateral acceleration. When flying at a relatively constant speed, the following acceleration constraints were sufficient:

$\forall k \in [0, \ldots, T-1]$, $\forall n \in [1, \ldots, N]$:

$$\ddot{x}(k)\sin(2\pi n/N) + \ddot{y}(k)\cos(2\pi n/N) \leq a_{lat} \tag{7.2}$$

for the double integrator model, and

$\forall k \in [1, \ldots, T-1]$, $\forall n \in [1, \ldots, N]$:

$$\begin{aligned} & (\dot{x}_{cmd}(k) - \dot{x}_{cmd}(k-1))\sin(2\pi n/N) \\ + \quad & (\dot{x}_{cmd}(k) - \dot{x}_{cmd}(k-1))\cos(2\pi n/N) \leq a_{lat}\Delta t \end{aligned} \tag{7.3}$$

for the reference velocity model. The lateral acceleration bound was set at $a_{lat} = 18.1$ ft^2/s, corresponding to a maximum turn rate of $\omega_{\max} = a_{lat}/v_{\min} = 2.6$ deg/s at $v_{\min} = 400$ fps.

When the velocity is allowed to change, however, these inequalities overestimate the available forward acceleration, which was limited to $a_{fwd} = 5.0$ ft^2/s. Therefore, to distinguish between forward and lateral acceleration, the following constraints were included, similar to inequalities (2.22-b):

$\forall k \in [0 \ldots T-1]$, $\forall n \in [1 \ldots N]$:

$$(\ddot{x}(k) + \alpha v(0)^{-1}\dot{y}(k))\sin(2\pi n/N) + (\ddot{y}(k) - \alpha v(0)^{-1}\dot{x}(k))\cos(2\pi n/N) \leq \beta \tag{7.4}$$

$$(\ddot{x}(k) - \alpha v(0)^{-1}\dot{y}(k))\sin(2\pi n/N) + (\ddot{y}(k) + \alpha v(0)^{-1}\dot{x}(k))\cos(2\pi n/N) \leq \beta \tag{7.5}$$

for the double integrator model, and

$\forall k \in [1, \ldots, T-1]$, $\forall n \in [1, \ldots, N]$:

$$\begin{aligned} & (\dot{x}_{cmd}(k) - \dot{x}_{cmd}(k-1) + \alpha v(0)^{-1} \dot{y}(k) \Delta t) \sin(2\pi n/N) \\ + \quad & (\dot{y}_{cmd}(k) - \dot{y}_{cmd}(k-1) - \alpha v(0)^{-1} \dot{x}(k) \Delta t) \cos(2\pi n/N) \leq \beta \Delta t \qquad (7.6) \\ & (\dot{x}_{cmd}(k) - \dot{x}_{cmd}(k-1) - \alpha v(0)^{-1} \dot{y}(k) \Delta t) \sin(2\pi n/N) \\ + \quad & (\dot{y}_{cmd}(k) - \dot{y}_{cmd}(k-1) + \alpha v(0)^{-1} \dot{x}(k) \Delta t) \cos(2\pi n/N) \leq \beta \Delta t \qquad (7.7) \end{aligned}$$

for the reference velocity model. Here, $v(0)$ is the current absolute ground speed, $\alpha = (a_{lat}^2 - a_{fwd}^2)/(2a_{fwd}) = 30.4$ fps^2/s, and $\beta = \sqrt{\alpha^2 + a_{lat}^2} = 35.4$ fps^2/s. As discussed in Section 3.4, these inequalities describe the intersection of two circles in which the inertial acceleration vector must lie. The short axis of this intersection has length $2a_{fwd}$ and is aligned with the velocity vector at the first time step. The long axis captures the larger lateral acceleration bound and has length $2a_{lat}$. As such, the intersection approximates the dynamically feasible acceleration profile at the initial time step.

Feasibility Constraints

The demonstration only considered rectangular no-fly zones aligned with the east-north coordinate frame, using constraints (2.30) in 2D to guarantee obstacle avoidance. To prevent the UAV from cutting corners, the actual obstacles were enlarged with a safety boundary of $d_{safe} = v_{\max} \Delta t / \sqrt{2} \approx 3200$ ft. Feasibility was guaranteed by ensuring that either a left or right loiter circle lying inside the circular detection region did not intersect with any of the obstacles. As detailed by inequalities (5.9)-(5.10), sample points along both circles were expressed as affine functions of the last state $\boldsymbol{x}(T)$ in the planning horizon. Again, because of the sampling procedure, the obstacles were enlarged on all sides by a thickness d_{loiter}. The demonstration used 8 sample points, which, given a maximum turn radius of 1.9 mi, resulted in $d_{loiter} = 0.6$ mi.

Cost Function

The objective was to guide the UAV between waypoints in the fastest possible way. The exact shortest time between two states, however, can only be computed if the planning horizon spans that arrival time, or if an exact cost-to-go is known. Since in the scenario of interest the environment was not characterized beyond a certain detection radius around the vehicle, computing an exact time-to-go function as proposed in [7] and [68] was not possible. Instead, the following heuristic was used.

If there are no known obstacles intersecting the straight line between the waypoint and the current location, that waypoint is used as the desired state in the cost function. In case there are obstacles blocking this direct line of sight, the shortest path (as far as the known obstacles are concerned) must go through one of the visible corner points of these no-fly zones. This point can then act as an intermediate waypoint en route to the final destination. To determine this optimal intermediate point, a grid is constructed between the corner points of all known obstacles interfering with the line of sight. A shortest path algorithm is then run to compute the approximate shortest time towards the goal from each visible corner point, thereby assuming that the UAV is flying at maximum speed. As such, the "best" intermediate waypoint is determined by minimizing the total time from the current location to one of the visible vertices and from that vertex to the destination as given by the approximate cost-to-go function.

Using this intermediate (or, in the obstacle-free case, the original) waypoint $\mathbf{p}_f = [x_f \; y_f]'$, the piecewise linear cost function

$$\min J = \sum_{k=0}^{T} -q\boldsymbol{v}(k)'(\boldsymbol{p}_f - \boldsymbol{p}_{estim}) + r|\boldsymbol{p}(k) - \boldsymbol{p}_f| \tag{7.8}$$

was used to design a *fast* trajectory between the initial position $\boldsymbol{p}(0) = \boldsymbol{p}_{estim} = [x(0) \; y(0)]'$ in the planning horizon and $\boldsymbol{p}_f$. Similarly to cost functions (3.35) and (5.12), the first term in this objective function tries to maximize the scalar product of the inertial velocity $\boldsymbol{v}(k) = [\dot{x}(k) \; \dot{y}(k)]'$ with the vector that is pointing from the initial position to the desired one, speeding the aircraft up to its maximal velocity and turning it towards waypoint $\boldsymbol{p}_f$.

If, at a certain iteration, the planned trajectory passes through or near waypoint $\boldsymbol{p}_f$ at a time step $T_f \leq T$ in the planning horizon, the cost function for the next iteration is split into two parts:

$$\begin{aligned} \min \tilde{J} \;\; &= \;\; \sum_{k=0}^{T_f-1} -q_f\boldsymbol{v}(k)'(\boldsymbol{p}_f - \boldsymbol{p}_{estim}) + r_f|\boldsymbol{p}(k) - \boldsymbol{p}_f| \\ &+ \;\; \sum_{k=T_f}^{T} -q_n\boldsymbol{v}(k)'(\boldsymbol{p}_n - \boldsymbol{p}_f) + r_n|\boldsymbol{p}(k) - \mathbf{p}_n| \end{aligned} \tag{7.9}$$

in which $\boldsymbol{p}_n$ is the next (intermediate) waypoint. The first $T_f - 1$ time steps are thus used to minimize the cost towards waypoint $\boldsymbol{p}_f$; the remaining steps aim at minimizing the cost towards the next waypoint $\boldsymbol{p}_n$. As a result, depending on the relative weighting, the MILP optimization will produce a trajectory that passes through or close by $\boldsymbol{p}_f$ and aims for $\boldsymbol{p}_n$ next. The SEC demonstration used $T = 6$, corresponding to an effective planning length of 1 minute, and all weights in (7.8) and (7.9) were set to 1.

7.5.2 Implementation

The Trajectory Generation Module was implemented in C++ and ILOG's Concert Technologies. To interface with the UAV avionics and guarantee hard real-time execution, it was integrated with Boeing's Open Control Platform (OCP) [106]. The software ran on a Pentium 4 Linux laptop with 2.4 GHz clock speed that was mounted in the aircraft, and interacted with the UAV avionics through a set of pre-defined command variables. Through the OCP interface the laptop received GPS, ground speed and turn rate data, among other, at a rate of 20 Hz. The guidance module itself, however, only ran at 1 Hz. It consisted of three subroutines: a pre-processing step, an optimization step, and a post-processing step, which are now discussed in more detail.

Pre-Processing

The pre-processing routine was called every second and determined all parameters of the MILP problem. It subsequently 1) selected the correct LTI model, 2) estimated the initial state for the current planning horizon, 3) determined the relevant obstacles, 4) enlarged the obstacles with the appropriate safety envelope, 5) determined the intermediate waypoint, and 6) selected the appropriate cost function. In addition, for numerical stability purposes and to speed up the MILP optimization, all latitude/longitude position data and obstacle

coordinates were transformed to an east-north axis frame in kilometer units with the current position of the aircraft as the origin. The ground velocity of the aircraft was scaled to km/s accordingly.

Optimization

The optimization step was implemented using the ILOGs Concert Technologies, a C++ based modeling language for use with CPLEX. It enables one to encode a large MILP problem in a compact form that is similar to the mathematical representation of it. To guarantee hard real-time execution of the mission software, we again used CPLEX's optional limit on computation time, which was set to 0.85 s. After the allocated time has passed, CPLEX then either returns a feasible solution within a predefined optimality gap (set to 10^{-4}), a feasible solution outside the optimality gap, or no solution at all. The last situation occurs when the MILP itself is infeasible or when no feasible solution can be found in time (e.g., because the problem is too complex).

Ideally, by definition of the loiter circle constraints, the trajectory planning problem remains feasible at all times. However, because of disturbances such as wind gusts, the initial velocity might fall outside the constraint bounds or the vehicle might be blown off course to a position from where an obstacle-free MILP solution no longer exists. The first situation is easy to spot and can be resolved ahead of time by resetting the velocity bounds in the pre-processing step. Infeasibilities caused by obstacles, however, are harder to predict and resolve. In that case, the UAV should resort to its backup plan, consisting of the remaining time steps and loiter circle of the previous plan.

If the control authority used in the MILP problem (i.e., the admissible acceleration and turn rate limit) is somewhat conservative w.r.t. the actual performance of the vehicle, robust trajectories can be designed. Then, in case the UAV gets blown off course to a state from which no feasible solution to the MILP exists, the aircraft can use its additional control authority to get back to feasibility within a few time steps [71]. Our code therefore used maximum acceleration and turn rate bounds that are smaller than the actual ones available to the waypoint controller. As a result, sequences of more than two infeasible receding horizon iterations never occurred.

Although the pre-processing step was repeated every second, the optimization function was nominally only executed every 10 s: the $\sim$10 s time constant of the T-33 made a higher planning rate unnecessary. Only when a large disturbance or an additional obstacle was detected, or when the vehicle was in backup plan mode (i.e., when the last MILP problem was infeasible), was the optimization routine executed at the next second. This way the available time slots could be occupied by computations required by the Task Scheduling and Natural Language Interface components.

Post-Processing

The post-processing routine performed the feasibility check by interpreting a CPLEX flag and updated the current trajectory (i.e., the current waypoint list), the loiter direction and a backup plan waypoint counter accordingly. In the nominal case where a feasible solution was found, the variables of interest were the 6 new states of the planning horizon (i.e., the new waypoint coordinates with corresponding velocity vectors) and the new loiter direction. The coordinates were first transformed back to the original Greenwich-referenced longitude and latitude axis frame and the velocity was rescaled to fps. Next, the old plan was flushed

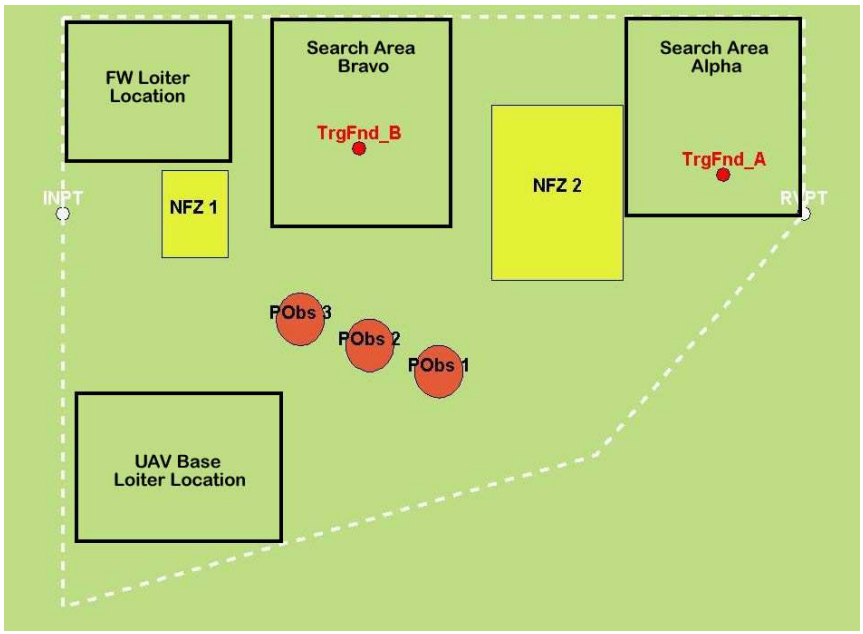


Figure 7-4: Sample scenario map for the MIT SEC Capstone Demonstration. The flight area is approximately 40 mi long along the northern boundary.

and replaced by the new one. The backup plan counter was set to 1, pointing to the first entry in the waypoint/state list, which was then given to a waypoint controller that issued forward velocity and turn rate commands to the vehicle.

If no feasible solution was found, however, the remainder of the previous trajectory was used as a backup plan. In that case, the backup plan counter was increased by one to point to the next waypoint of the existing plan. If the counter exceeded 6, depending on the value of the loiter direction binary, the left or right loiter circle was initiated by issuing a "Set and Hold Turn Rate" command to the UAV. Its value was set to the maximum available turn rate at the current velocity, e.g., 3 deg/s at 400 fps, which was slightly more aggressive than the maximum 2.6 deg/s accounted for in the planning problem. The turn command thus resulted in a smaller loiter circle than planned, which introduced some robustness to perturbations along the trajectory. As long as the vehicle remained in the backup plan mode, the MILP optimization was executed every second (but the counter only updated every 10 s) until a new feasible plan was found.

7.6 Simulation and Flight Test Results

Using the narrative outlined in Section 7.2, various sample scenarios were designed that are depicted in Figure 7-4. The flight area is approximately 40 mi across (east to west along the northern boundary) and 30 mi wide (north to south along the western boundary). There are two pre-determined no-fly zones (listed as "NFZ 1" and "NFZ 2") and three potential pop-up threats (denoted by "PObs 1," "PObs 2," and "PObs 3"), which can be activated during flight. In addition, there are two mission task areas (labeled "Search Area Alpha" and "Search Area Bravo"). Each task area has three potential ingress conditions which can be selected by the FW WSO before the vehicle reaches the task area location. Each task

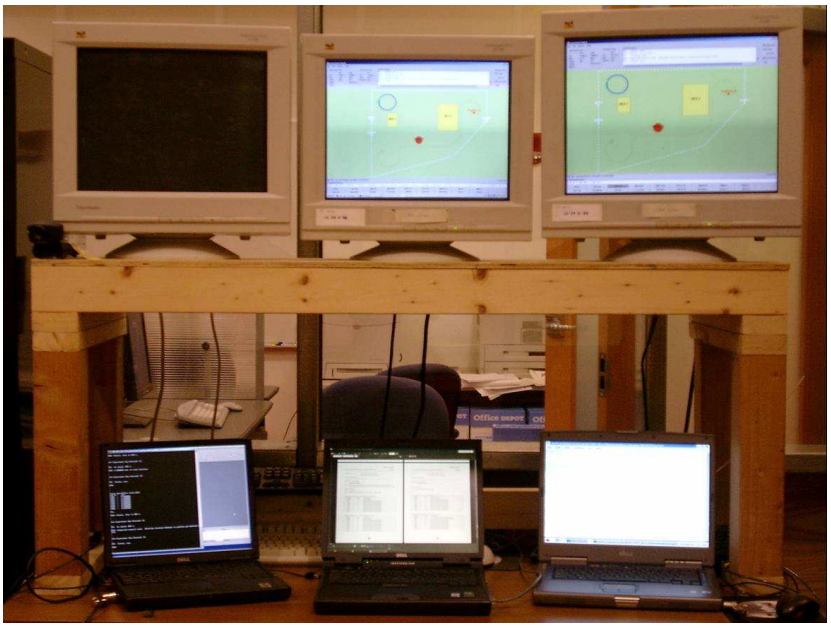


Figure 7-5: SEC demonstration Simulation-In-the-Loop (SIL) laboratory setup

area also includes a threat/target (denoted by "TrgFnd A" and "TrgFnd B") which the UAV searches for and locates during the mission. Finally, the UAV starts the mission from the UAV Base Loiter location in the southwest corner of the flight area, and the FW vehicle maintains a loiter pattern near the northern border until the target has been detected.

7.6.1 Simulation Results

To aid in the development of our guidance system, a real-time Simulation-In-the-Loop (SIL) test platform was built at MIT, which is shown in Figure 7-5. Besides the OCP, it included Boeing's DemoSim vehicle simulations for the UAV (T-33) and FW (F-15) aircraft, which were executed on separate computers with a Link-16 communication interface. The mission system software ran on two laptops similar to the ones mounted in the aircraft during the flight-test experiments. Using wireless ethernet connections through the laboratory LAN, command latency and other real-time issues could be simulated. Besides communications link latency, test conditions included message drop-outs, invalid experiment key selection, data scaling issues and modeling errors.

Figure 7-6 shows one of the many initialization tests. As the FW aircraft and UAV approach the flight area, the demonstration software is initialized: the UAV automatically flies to the UAV Base Loiter Location, where it remains until it is commanded another task. The loiter task itself is defined as a series of six waypoints with a fixed ingress location and heading. As can be seen from the picture, the UAV successfully avoids NFZ1 while flying to the loiter area.

Next, Figure 7-7 shows one of the pop-up obstacle avoidance tests used to verify the feasibility guarantees of the MILP trajectory planning algorithm. In this test, two pop-up obstacles were placed into the demonstration area as the UAV was en route to Search Area Alpha. The resulting trajectory highlights the MILP algorithm's ability to maintain dynamically feasible, obstacle-free paths for the vehicle after unexpected changes to the

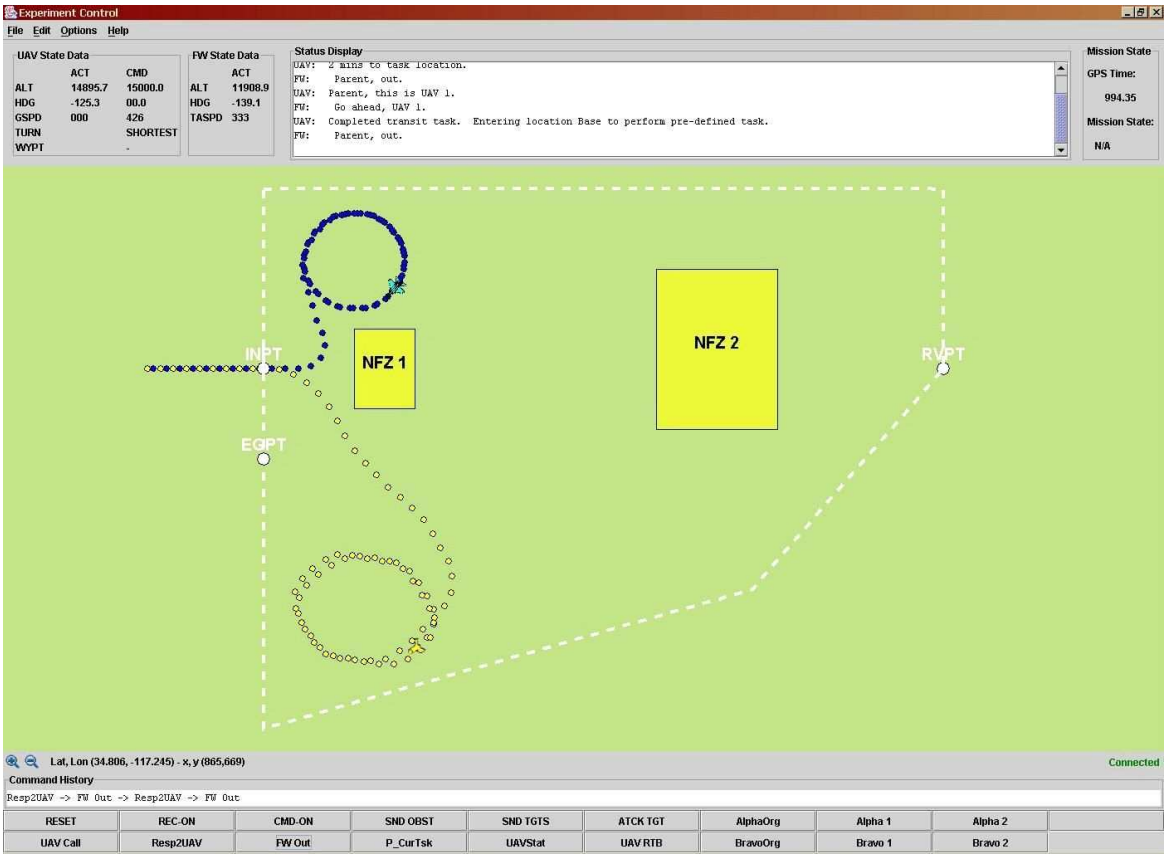


Figure 7-6: SIL Test 1 - Initialization of the SEC demonstration: the UAV (in light) enters a loiter pattern.

environment right outside its detection radius. For example, when the UAV is south of the first pop-up obstacle (PObs 3), the second one (PObs 1) is inserted, causing the UAV to immediately turn left and proceed northeast over it. After passing the pop-up obstacles, the vehicle levels out and flies at a safe distance from No-Fly Zone 2 (NFZ 2) before turning north to enter Search Area Alpha to perform a search task.

Figure 7-8 depicts a test where the UAV was commanded to fly two consecutive missions from the UAV Base Loiter Location. The main objective of this test was to ensure that the vehicle returns to its loiter location after it finishes a certain task (provided that another task was not given). First, the FW WSO commands the UAV to proceed to one of the search areas, but does not issue the Return-To-Base (RTB) command during the mission. Still, after the UAV finishes its search of the task area, it informs the FW WSO that it has completed the search task and will proceed back to the Base Loiter Location to await another set of commands. This test shows that the software provides the vehicle operators and test directors with flexibility in task and mission management during flight.

Figure 7-8 also shows the UAV's coverage over both search areas during the search task portions of the mission. Notice that the two search patterns are almost identical: the same task defining waypoint sequence (relative to the ingress position of the task area) was used in both search tasks, showing that the MILP guidance approach can accurately track waypoint plans. In addition, the UAV safely avoids two pop-up obstacles en route to each search area.

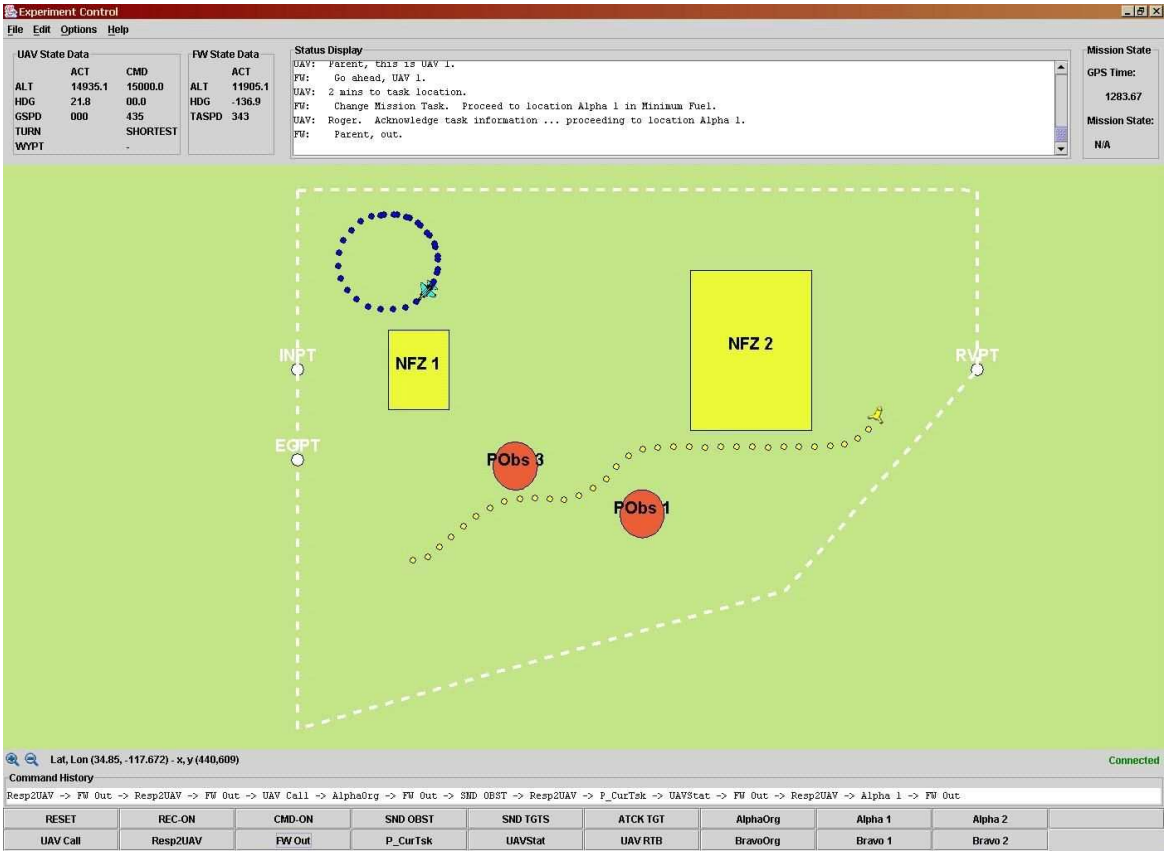


Figure 7-7: SIL Test 2 - Pop-up obstacle test: the UAV safely avoids both pop-up threats.

7.6.2 Flight-Test Results

From mid-April 2004 to mid-June 2004, the final demonstration software was turned over to Boeing Phantom Works in St. Louis for verification and validation testing on a hardware in the loop simulator. After successfully completing this step, the software was transitioned to the actual vehicle setup for testing at NASA Dryden in late June 2004. Figure 7-9 shows a system level diagram of the flight experiment setup. During the test, the main role of the T-33's two person crew was to fly the vehicle to the demonstration area, activate the demonstration software and manage the vehicle in the event of failures. In addition, for technical reasons, the T-33 pilot executed the forward velocity commands produced by the MILP guidance algorithm. The turn rate, however, was directly commanded by the laptop.

Although the FW WSO was only able to select UAV-tasks from a pre-defined list of options, the actual mission was not pre-planned. The T-33's rear-seat operator (nicknamed the "Guy-In-Back" or "GIB") observed the progress of the demonstration and a Ground Station Operator (GSO) added pop-up obstacles via experiment key commands. Although the possible locations of the mission task areas and pop-up obstacles were pre-defined, they were selected randomly in real-time, thus introducing another degree of uncertainty into the flight experiment. The Test Coordinator (TC) monitored the demonstration from the ground station and communicated status information about the local airspace to the pilots.

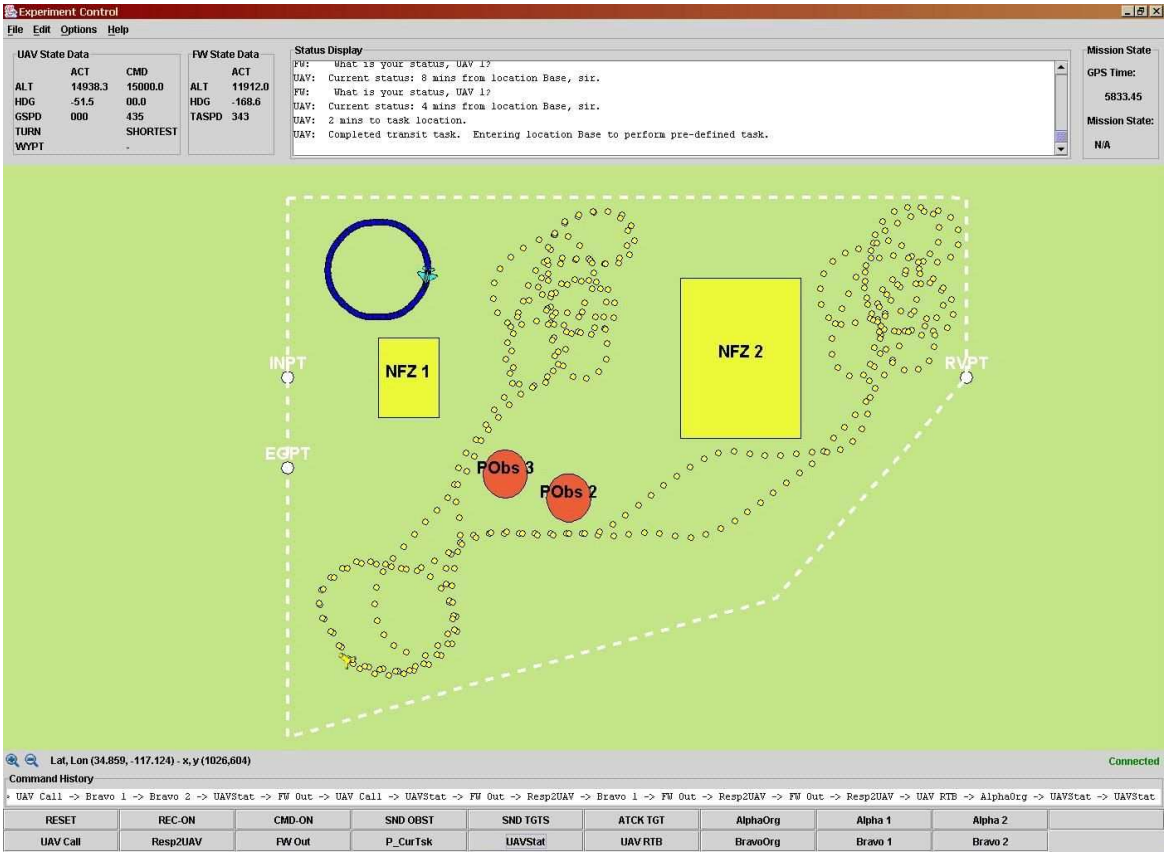


Figure 7-8: SIL Test 3 - Simulated flight with two consecutive search missions.

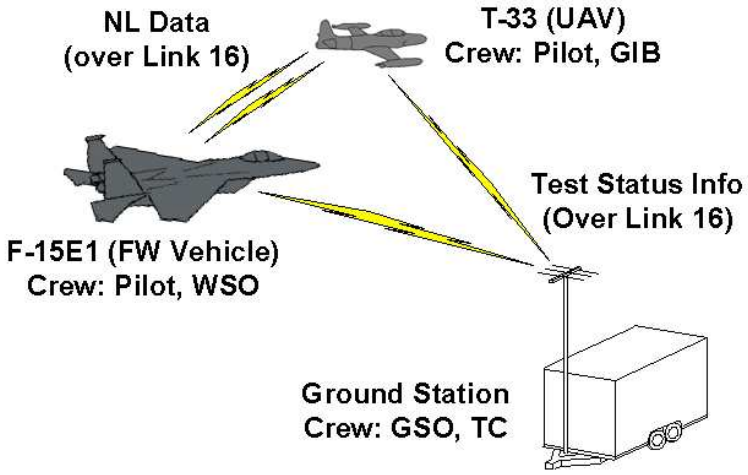


Figure 7-9: Flight experiment system level diagram

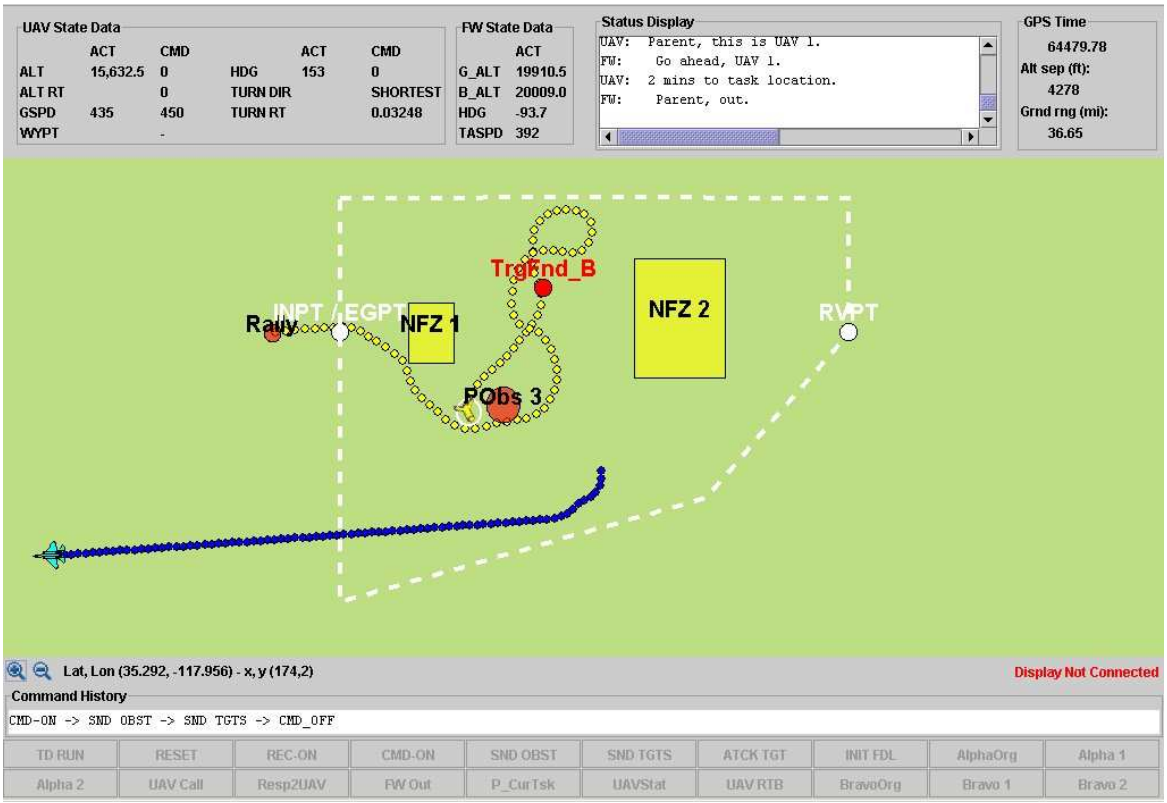


Figure 7-10: SEC Flight-Test: UAV/T-33 (in light) with simulated F-15 (in dark)

Flight-Test with T-33/UAV and Simulated F-15

In the first test, depicted in Figure 7-10, the T-33/UAV flew a mission with a simulated F-15 and the GSO issuing the natural language commands. The flight took place in the morning of Thursday, June 17th, 2004, with a wind of 5 to 10 knots blowing from the southwest corner of the flight area (at a heading of 220 to 240 degrees).

The UAV started west of the ingress point (labelled "INPT / EGPT" in Figure 7-10) with a heading of 090. After the GIB initialized the mission software, the UAV began turning south to avoid NFZ 1. After it passed the lower left hand corner of NFZ 1, the Test Coordinator notified the T-33/UAV operators that the southwestern corner of the test area had to be avoided because of unplanned flight activity there. As such, when the vehicle was approximately two miles SSW of NFZ 1, the GSO commanded the UAV to proceed to Task Area Bravo before reaching the UAV Base Loiter Location. The UAV responded and began turning left toward it. This verified the flexibility of the mission software and the ability of an operator to easily change the tasks in real-time.

Within two minutes of the last command, the GSO inserted Pop-up Obstacle 3 into the test area. At this point, the vehicle had a heading between 070 and 080 and was approximately four miles from the obstacle. It began to turn right to avoid the obstacle and flew along its southern boundary, successfully avoiding it even though the obstacle was inserted inside the vehicle's detection. After passing it, the UAV proceeded to turn north toward Task Area Bravo, initiated the search pattern, and notified the GSO of its status. As the vehicle was facing SSE, it was near the target location and the GSO inserted the target into

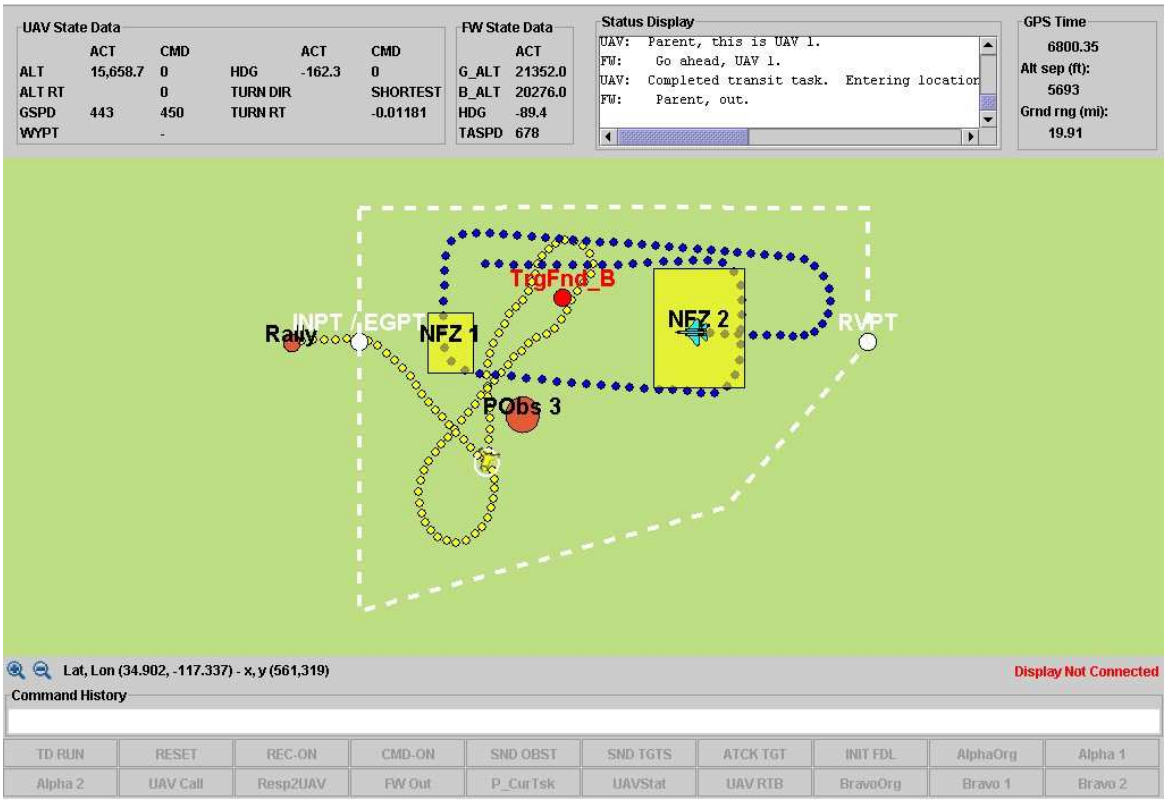


Figure 7-11: SEC Flight-Test: UAV/T-33 (in light) and actual F-15 (in dark) providing natural language commands.

the environment. The UAV sent the proper status messages to the operator, after which the GSO commanded it to return to base. The vehicle began to turn right, safely flew southwest around Pop-up Obstacle 3, and sent a "Two Minutes to Task Location" notification. Since the southwest airspace was off-limits, however, the UAV was not permitted to fly to its Base Loiter Location, and the mission was ended.

The flight test marked the first time a MILP-based onboard guidance system operating in real-time was used to control a UAV. The natural language software effectively communicated mission status to the GSO and was successfully used to command the UAV to perform and change tasks during the flight. The GSO remarked that he found the interface easy to use and helpful in following the progress of the mission.

Flight-Test with T-33/UAV and Actual F-15

In a second test, shown in Figure 7-11, the T-33/UAV flew a successful mission with the F-15 WSO issuing the natural language commands. This flight took place in the afternoon on Wednesday, June 23rd, 2004, with a SW wind (220-240 deg) between 10 and 15 knots. The UAV again started west of the ingress point with a heading of 090 and began turning south to avoid NFZ 1 after the GIB started the experiment software. After passing the no-fly zone, the vehicle steadily moved to the UAV Base Loiter Location. When it had a heading of 270, the F-15 WSO commanded the UAV to fly to Task Area Bravo. It responded and began turning northward. When the GSO inserted Pop-up Obstacle 3, the UAV notified

the F-15 that it detected the threat and successfully avoided the obstacle. As it approached the task area, the UAV informed the F-15 WSO that it was two minutes from the ingress point it was initially given. At that time, the WSO commanded the UAV to change the entrance location. It responded and proceeded to the new ingress point. This again verified the flexibility of the task scheduling and trajectory planning software.

After reaching the task area, the UAV notified the F-15 WSO and began its search pattern. As it turned left toward the southern boundary of the search area, the GSO inserted Target B into the environment. The UAV notified the F-15 WSO and sent an image. The F-15 WSO then commanded the UAV to return to base. The UAV acknowledged the command and flew back to its loiter location, avoiding obstacles and providing status notifications to the F-15 WSO along the way. The demonstration was ended when the UAV successfully returned to the UAV Base Loiter Location.

This flight-test marked the first time that a natural language interface was used by a manned vehicle to task and command a UAV in real-time. The F-15 WSO also remarked that he found the interface easy to use and helpful in following the progress of the mission. In addition, the test marked the first time that an onboard MILP-based guidance system was used to control a UAV in coordination with a manned vehicle. Furthermore, it was the first successful in-flight cooperation demonstration between the F-15 and UAV for the Boeing team. Overall, both flight-tests provided an important proof-of-concept of the capabilities of the MIT mission software as previously witnessed in the laboratory: the flexibility of the task scheduling software allowed the test team to make adjustments in real-time, while the trajectory generation algorithm safely guided the UAV through the environment.

7.7 Conclusion

This chapter described the development, architecture and testing of a manned vehicle - UAV mission system that allows an operator in a manned aircraft to issue mission level commands to an autonomous aircraft in real-time. A natural language interface was presented that allows the manned and unmanned vehicle to communicate in languages understood by both agents. A task scheduler transformed these commands into a dynamic mission plan consisting of task waypoints. The latter were then given to a real-time MILP-based trajectory planner, generating obstacle-free trajectories through a partially-known environment by implementing the feasible receding horizon strategy presented in Chapter 4.

The complete mission system was successfully tested using high-fidelity software- and hardware in the loop simulations. Actual flight-test results with an F-15 and an autonomous T-33 were presented that provided an important proof-of-concept of the benefits and real-time capabilities of the natural language interface and MILP-based guidance methodology. After these first validation steps, the approach and algorithms are ready to being transitioned to larger problems, such as platforms with multiple unmanned vehicles running distributed task assignment and trajectory planning strategies. The eventual goal is to have a single operator issue team level mission commands using natural language. We believe that in the future natural language interfaces will be the most efficient way to communicate with unmanned vehicle systems.

Chapter 8

Conclusion

8.1 Summary

This thesis presented several new concepts and algorithms for online receding horizon trajectory planning of autonomous vehicles. We started off with a general problem formulation and used mixed-integer linear programming as the modeling and implementation framework. A new, more efficient approach to capture avoidance constraints for arbitrarily shaped obstacles was introduced. Next, the basic linear dynamics used in Chapter 2 were extended in Chapter 3 to incorporate agile nonlinear behavior into the planning formulation. A hybrid control architecture for trajectory planning of agile vehicles was presented that combines multiple velocity control modes with a maneuver scheduler. The former provide the flexibility to precisely navigate between waypoints in a cluttered environment, while the latter enables execution of pre-programmed maneuvers at the limit of the vehicle capabilities. The closed-loop dynamics under this control architecture were described by a simple hybrid model consisting of a set of LTI modes and discrete, fixed-duration state transitions. Online trajectory optimization through cluttered environments was again formulated using MILP. The approach was worked out in detail for a small-scale helicopter model based on MIT's aerobatic X-Cell.

Chapter 4 then introduced the concept of a feasible invariant set that is a periodic, dynamically feasible and obstacle-free sequence of states in which a vehicle can remain for an indefinite period of time. Constraining all intermediate receding horizon trajectories to terminate in such sets guarantee feasibility of the planning problem at all future iterations. These terminal feasible invariant sets were expressed as a collection of affine transformations of the last state in the planning horizon and as a result were computed online as part of the optimization problem. An explicit formulation for a UAV was worked out using obstacle-free loitering circles as feasible invariant sets. Next, safety was defined as being able to leave the terminal feasible invariant set at some time step in the future and backtrack along the executed trajectory. A receding horizon formulation that maintains safety and feasibility at all time steps was presented. A practical approach that only maintains safety was discussed as well.

In Chapter 5, the terminal feasible invariant set principle was used to design a distributed algorithm for multi-vehicle path planning with feasibility guarantees. A receding horizon planning strategy was adopted for each vehicle individually that accounts for the trajectories of the neighboring ones. Within one iteration of the algorithm, the vehicles update their trajectories in a sequential fashion. Feasibility at all times is guaranteed by constraining

the intermediate plans of all vehicles to terminate in individual feasible invariant sets. A MILP-based formulation for multiple aircraft using loiter circles was worked out in detail.

Chapter 6 presented a proof-of-concept application of online MILP-based trajectory planning for multiple vehicles. The problem of interest was to maintain line of sight communication in a cluttered environment between a mission vehicle and ground station using a relay network. Both a centralized and a distributed formulation were given. The former was successfully implemented on two X-Cell helicopters and tested in a real-world scenario.

Chapter 7 then presented UAV flight-test results of the feasible receding horizon planning strategy of Chapter 4. It discussed the development, architecture and testing of a manned vehicle - UAV mission system that allows an operator in a manned aircraft to issue mission level commands to an autonomous aircraft in real-time. The complete mission system was successfully tested using a manned F-15 and an autonomous T-33 aircraft. The results in both chapters showed that receding horizon MILP-based trajectory planning is a flexible framework and a feasible option for real-time guidance.

8.2 Future Work

The concepts introduced in this thesis give rise to various topics for future research. First, throughout the thesis, we only considered the nominal planning problem which does not account for external disturbances acting on the vehicle or uncertainties in the dynamic models. Introducing robustness in the planning problem is therefore a necessary next step. Among other, interesting problems related to robustness include handling uncertainties in the location of obstacles and perturbations during maneuver execution of the LTI-MA, formulating robustly feasible invariant sets leading to robust feasibility and safety, and tackling uncertainties in the execution of the individual plans in the distributed planning algorithm of Chapter 5. Some initial work on the last two issues was recently begun in [71] and [70].

A second broad topic is related to the performance and stability of the various planning formulations. For example, good cost-to-go functions for the LTI-MA that account for the availability of maneuvers need to be developed. This may include functions that are computed offline and then incorporated in the online trajectory optimization. Functions for often recurring situations, such as turning around a corner or reversing direction, or functions corresponding to different planning modes as discussed in Chapter 3 could be designed. Good cost-to-go functions are required to prevent the vehicle from getting trapped in local minima. For the distributed multi-vehicle case, a related performance and stability concern is the possible existence of live-lock situations, i.e. the condition when all vehicles are trapped in their respective terminal feasible invariant sets. Proving that a particular objective function or algorithm will be able to avoid such situations is another problem that has to be tackled. Furthermore, comparing the performance of the distributed approach to centralized formulations with respect to the quality of the resulting trajectories and the time required to compute them would be another interesting topic of future work.

Third, it may be worthwhile to investigate the specific structure of the MILP formulation for path planning in order to find specific heuristics and cuts in the MILP solution algorithms. Rather than fully relying on CPLEX to analyze the matrix structure and apply particular heuristics or algorithms, guiding the optimization engine by exploiting a priori knowledge of the problem structure could speed up the computation. Alternatively, combining the MILP approach with alternative approaches such as constraint programming

could take some of the computational load away as well.

Finally, further implementation of the concepts developed in this thesis and testing of various applications would be desirable. This includes actual helicopter flight-tests of the LTI-MA trajectory planning framework, possibly with the addition of flexible maneuvers as introduced in [31], and demonstrations of the safe distributed algorithm for both homogeneous and heteregenous vehicle teams. Work on the latter is currently already being undertaken at MIT.

Bibliography

[1] M. Athans and P. Falb. *Optimal Control, An Introduction to the Theory and Its Applications*. McGraw-Hill, 1966.

[2] J. Barraquand, L.E. Kavraki, J.C. Latombe, T.Y. Li, R. Motwani, and P. Raghavan. A random sampling scheme for path planning. *International Journal of Robotics Research*, 16(6):759–774, 1997.

[3] J. Barraquand, B. Langlois, and J.-C. Latombe. Numerical potential field techniques for robot path planning. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(2):224–241, March/April 1992.

[4] A. Bayen, E. Cruck, and C. Tomlin. Guaranteed overapproximations of unsafe sets for continuous and hybrid systems: Solving the Hamilton-Jacobi equation using viability techniques. In C. Tomlin and M. Greenstreet, editors, *Hybrid Systems: Computation and Control*, volume 2289 of *Lecture Notes in Computer Science (LNCS)*, pages 90–104. Springer Verlag, 2002.

[5] R. Beard and T. McLain. Multiple UAV cooperative search under collision avoidance and limited range communication constraints. In *Proc. 42nd IEEE Conference on Decision and Control*, Maui, HI, December 2003.

[6] R. Beard, T. McLain, M. Goodrich, and E. Anderson. Coordinated target assignment and intercept for unmanned air vehicles. *IEEE Transactions on Robotics and Automation*, 18(6):911–922, 2002.

[7] J. Bellingham, A. Richards, and J. How. Receding horizon control of autonomous aerial vehicles. In *Proc. 2002 American Control Conference*, pages 3741–3746, Anchorage, AK, May 2002.

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

[9] A. Bemporad, F. Borrelli, and M. Morari. Piecewise linear optimal controllers for hybrid systems. In *Proc. 2001 American Control Conference*, volume 2, pages 1190–1194, Chicago, IL, June 2001.

[10] A. Bemporad, F. Borrelli, and M. Morari. Model predictive control based on linear programming: the explicit solution. *IEEE Transactions on Automatic Control*, 47(12):1974–1985, December 2002.

[11] A. Bemporad, A. De Luca, and G. Oriolo. Local incremental planning for a car-like robot navigating among obstacles. In *Proc. 1996 IEEE Int. Conference on Robotics and Automation*, pages 1205–1211, Minneapolis, MN, April 1996.

[12] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, 1999.

[13] A. Bemporad and M. Morari. Robust model predictive control: A survey. In A. Garulli, A. Tesi, and A. Vicino, editors, *Robustness in Identification and Control*, volume 245 of *Lecture Notes in Control and Information Sciences*, pages 207–226. Springer Verlag, 1999.

[14] A. Bemporad, M. Morari, V. Dua, and E.N. Pistikopoulos. The explicit solution of model predictive control via multiparametric quadratic programming. In *Proc. 2000 American Control Conference*, pages 872–876, Chicago, IL, June 2000.

[15] C. Bererton, G. Gordon, S. Thrun, and P. Khosla. Auction mechanism design for multi-robot coordination. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Proc. 2003 Conference on Neural Information Processing Systems (NIPS)*. MIT Press, 2003.

[16] D.P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 2nd edition, 1999.

[17] D.P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 1. Athena Scientific, Belmont, MA, 2nd edition, 2000.

[18] D. Bertsimas and J. N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, Belmont, MA, 1997.

[19] D. Bertsimas and R. Weismantel. *Optimization Over Integers*. Dynamic Ideas, Belmont, MA, May 2005.

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

[21] R. Bohlin and L.E. Kavraki. Path planning using lazy PRM. In *Proc. 2000 IEEE Int. Conference on Robotics and Automation*, pages 521–528, San Francisco, CA, April 2000.

[22] F. Borrelli. *Constrained Optimal Control of Linear and Hybrid Systems*, volume 290 of *Lecture Notes in Control and Information Sciences*. Springer-Verlag, Berlin Heidelberg, May 2003.

[23] D. Brelaz. New methods to color the vertices of a graph. *Comm. ACM*, 22:251–256, 1979.

[24] A.E. Bryson and Y.C. Ho. *Applied Optimal Control: Optimization, Estimation, and Control*. Hemisphere Publishing Corp., New York, 1975.

[25] S. Butenko, R. Murphey, and P. Pardalos (Eds.). *Recent Developments in Cooperative Control and Optimization*. Kluwer Academic Publishers, Norwell, MA, 2003.

[26] J. Canny. *The Complexity of Robot Motion Planning*. MIT Press, Cambridge, MA, 1988.

[27] P.R. Chandler, M. Pachter, and S. Rasmussen. UAV cooperative control. In *Proc. 2001 American Control Conference*, Arlington, VA, June 2001.

[28] 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 *Proc. 2002 American Control Conference*, Anchorage, AK, May 2002.

[29] H. Chen and F. Allgower. A quasi-infinite horizon nonlinear model predictive scheme with quaranteed stability. *Automatica*, 14(10):1205–1217, 1998.

[30] H. Choset. *Sensor Based Motion Planning: the Hierarchical Generalized Voronoi Graph*. PhD thesis, California Institute of Technology, Pasadena, CA, 1996.

[31] C. Dever, B. Mettler, E. Feron, and J. Popovic. Trajectory interpolation for parametrized maneuvering and flexible motion planning of autonomous vehicles. In *Proc. AIAA Guidance, Navigation, and Control Conference*, number AIAA-2004-5143, Providence, RI, August 2004.

[32] W.B. Dunbar and R.M. Murray. Model predictive control of coordinated multi-vehicle formations. In *Proc. 41st IEEE Conference on Decision and Control*, pages 4631–4636, Las Vegas, NV, December 2002.

[33] C. A. Floudas. *Nonlinear and Mixed-Integer Programming - Fundamentals and Applications*. Oxford University Press, Oxford, UK, 1995.

[34] R. Fourer, D. M. Gay, and B. W. Kernighar. *AMPL, A modeling language for mathematical programming*. The Scientific Press, 1993.

[35] E Frazzoli. *Robust Hybrid Control for Autonomous Vehicle Motion Planning*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, May 2001.

[36] E. Frazzoli, M. A. Dahleh, and E. Feron. Real-time motion planning for agile autonomous vehicles. *AIAA Journal of Guidance, Control, and Dynamics*, 25(1):116–129, 2002.

[37] E. Frazzoli, Z.-H. Mao, J.-H. Oh, and E. Feron. Resolution of conflicts involving many aircraft via semidefinite programming. *AIAA Journal of Guidance, Control and Dynamics*, 24(1):79–86, 2001.

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

[39] V. Gavrilets. *Autonomous Aerobatic Maneuvering of Miniature Helicopters*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 2003.

[40] V. Gavrilets, E. Frazzoli, B. Mettler, M. Piedmonte, and E. Feron. Aggressive maneuvering of small autonomous helicopters: A human-centered approach. *International Journal of Robotics Research*, pages 795–807, October 2001.

[41] V. Gavrilets, I. Martinos, B. Mettler, and E. Feron. Aggressive maneuvering flight tests and simulation results of a miniature robotic helicopter. In *Proc. 8th International Symposium on Experimental Robotics*, Sant'Angelo d'Ischia, Italy, July 2002.

[42] V. Gavrilets, I. Martinos, B. Mettler, and E. Feron. Flight test and simulation results for an autonomous acrobatic helicopter. In *Proc. 21st Digital Avionics Conference (DASC)*, Irvine, CA, October 2002.

[43] V. Gavrilets, B. Mettler, and E. Feron. Nonlinear model for a small-size acrobatic helicopter. In *Proc. AIAA Guidance, Navigation, and Control Conference*, Montreal, Canada, August 2001. AIAA Paper 2001-4333.

[44] V. Gavrilets, B. Mettler, and E. Feron. Human-inspired control logic for automated maneuvering of miniature helicopter. *AIAA Journal of Guidance, Control and Dynamics*, 27(5), 2004.

[45] Google. Google earth. http://earth.google.com.

[46] J.R. Gossner, B. Kouvaritakis, and J.A. Rossiter. Stable generalized predictive control with constraints and bounded disturbances. *Automatica*, 33(4):551–568, 1996.

[47] 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., Hackensack, NJ, 2004.

[48] J. Hauser and A. Jadbabaie. Aggressive maneuvering of a thrust vectored flying wing: a receding horizon approach. In *Proc. 39th IEEE Conference on Decision and Control*, pages 3582–3587, Sydney, Australia, December 2000.

[49] J.E. Hopcroft, J.T. Schwartz, and M. Sharir. On the complexity of motion planning for multiple independent objects: PSPACE-hardness of the warehouseman's problem. *International Journal of Robotics Research*, 4(3):76–88, 1984.

[50] A. Howard, M.J. Mataric, and G.S. Sukhatme. An incremental self-deployment algorithm for mobile sensor networks. *Autonomous Robots, Special Issue on Intelligent Embedded Systems*, 13(2):113–126, 2002.

[51] D. Hsu, L.E. Kavraki, J-C. Latombe, R. Motwani, and S. Sorkin. On finding narrow passages with probabilistic roadmap planners. In *WAFR '98: Proceedings of the Third Workshop on the Algorithmic Foundations of Robotics on Robotics : the Algorithmic Perspective*, pages 141–153, March 1998.

[52] D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. *Int. Journal of Computational Geometry and Applications*, 9(4):46–512, 1996.

[53] J. Hu, M. Prandini, and S. Sastry. Optimal coordinated maneuvers for three-dimensional aircraft conflict resolution. *AIAA Journal of Guidance, Control and Dynamics*, 25(5):888–900, 2002.

[54] Y.K. Hwang and N. Ahuja. Gross motion planning - a survey. *ACM Comp. Surv.*, 24(3):219–291, 1992.

[55] G. Inalhan, D. Stipanovic, and C. Tomlin. Decentralized optimization, with application to multiple aircraft coordination. In *Proc. 41st IEEE Conference on Decision and Control*, Las Vegas, NV, December 2002.

[56] ILOG Inc. *ILOG CPLEX 9.0 User's guide*. Mountain View, CA, 2003.

[57] A. Jadbabaie. *Receding Horizon Control of Nonlinear Systems: A Control Lyapunov Function Approach*. PhD thesis, California Institute of Technology, Pasadena, CA, October 2000.

[58] F. Jean. Complexity of nonholonomic motion planning. *International Journal of Control*, 74(8):776–782, May 2001.

[59] N. Kalra, D. Ferguson, and A. Stentz. Hoplites: A market-based framework for planned tight coordination in multirobot teams. In *Proc. 2005 IEEE International Conference on Robotics and Automation*, Barcelona, Spain, April 2005.

[60] T. Karatas and F. Bullo. Randomized searches and nonlinear programming in trajectory planning. In *Proc.40th IEEE Conference on Decision and Control*, December 2001.

[61] L. Kavraki and J.-C. Latombe. Randomized preprocessing of configuration for fast path planning. In *Proc. 1994 IEEE International Conference on Robotics and Automation*, May 1994.

[62] L. Kavraki, P. Svestka, J. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. Technical report, Stanford University, Stanford, CA, 1994.

[63] L.E. Kavraki, F. Lamiraux, and C. Holleman. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[64] E.C. Kerrigan and J.M. Maciejowski. Invariant sets for constrained nonlinear discrete-time systems with application to feasibility in model predictive control. In *Proc. 39th IEEE Conference on Decision and Control*, Sydney, Australia, December 2000.

[65] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 1(5):90–98, 1986.

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

[67] A. Kott. *Advanced Technology Concepts for Command and Control*. Xlibris Corporation, Philadelphia, PA, 2004.

[68] Y. Kuwata and J. How. Stable trajectory design for highly constrained environments using receding horizon control. In *Proc. 2004 American Control Conference*, pages 902–907, Boston, MA, June 2004.

[69] Y. Kuwata and J. How. Three dimensional receding horizon control for UAVs. In *Proc. AIAA Guidance, Navigation, and Control Conference*, Providence, RI, August 2004. AIAA-2004-5144.

[70] Y. Kuwata, A. Richards, T. Schouwenaars, and J. How. Decentralized robust receding horizon control for multi-vehicle guidance. Submitted to 2006 American Control Conference.

[71] Y. Kuwata, T. Schouwenaars, A. Richards, and J. How. Robust constrained receding horizon control for trajectory planning. In *Proc. AIAA Guidance, Navigation, and Control Conference*, San Francisco, CA, August 2005.

[72] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Norwell, MA, 1991.

[73] J.-C. Latombe. Motion planning: a journey of robots, molecules, digital actors, and other artifacts. *International Journal of Robotics Research*, 18(11):1119–1128, November 1999.

[74] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006. Also available at http://msl.cs.uiuc.edu/planning/.

[75] S.M. LaValle and J.J. Kuffner. Randomized kinodynamic planning. In *Proc. IEEE International Conference on Robotics and Automation*, July 1999.

[76] T. Lemaire, R. Alami, and S. Lacroix. A distributed tasks allocation scheme in multi-UAV context. In *Proc. 2004 IEEE International Conference on Robotics and Automation*, pages 3622–3627, New Orleans, LA, April 2004.

[77] D. Leven and M. Sharir. An efficient and simple motion planning algorithm for a ladder moving in two-dimensional space amidst polygonal barriers. *Journal of Algorithms*, 8:192–215, 1987.

[78] F-L. Lian and R. Murray. Cooperative task planning of multi-robot systems with temporal constraints. In *Proc. 2003 IEEE Int. Conference on Robotics and Automation*, Taipei, Taiwan, September 2003.

[79] F.-L. Lian and R.Murray. Real-time trajectory generation for the cooperative path planning of multi-vehicle systems. In *Proc. 41st IEEE Conference on Decision and Control*, pages 3766–3769, Las Vegas, NV, December 2002.

[80] T. Lozano-Perez. *Spatial Planning with Polyhedral Models*. PhD thesis, Massachusetts Institute of Technology, June 1980.

[81] T. Lozano-Perez. Automaton planning of manipulator transfer movements. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(10):681–698, 1981.

[82] T. Lozano-Perez and M.A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Commun. ACM*, 22(10):560–570, 1979.

[83] J. Lygeros, C. Tomlin, and S. Sastry. Controllers for reachability specifications for hybrid systems. *Automatica*, 35(3), March 1999.

[84] N. Lynch, R. Segala, and F. Vaandrager. Hybrid I/O automata revisited. In M.D. Di Benedetto and A. Sangiovanni-Vincentelli, editors, *Hybrid Systems: Computation and Control*, volume 2034 of *Lecture Notes in Computer Science (LNCS)*, pages 403–417. Springer Verlag, 2001.

[85] J.M. Maciejowski. *Predictive Control with Constraints*. Prentice Hall, 2002.

[86] A. Makhorin. GLPK (GNU Linear Programming Kit). Available at http://www.gnu.org/software/glpk/glpk.html.

[87] A. Marigo and A. Bicchi. Steering driftless nonholonomic systems by control quanta. In *Proc. 37th IEEE Conference on Decision and Control*, December 1998.

[88] K. Marti and S. Qu. Path planning for robots by stochastic optimization methods. *International Journal of Intelligent and Robotic Systems*, September 1997.

[89] I. Martinos, T. Schouwenaars, J. DeMot, and E. Feron. Hierarchical cooperative multi-agent navigation using mathematical programming. In *Proc. 41th Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, October 2003.

[90] D.Q. Mayne. Control of constrained dynamic systems. *European Journal of Control*, 7(2-3):87–99, 2001.

[91] D.Q. Mayne, J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, June 2000.

[92] C.R. McInnes. Potential function methods for autonomous spacecraft guidance and control. In *Proc. AAS/AIAA Astrodynamics Conference*, pages 2093–2109, February 1995.

[93] B. Mettler, M. Valenti, T. Schouwenaars, E. Frazzoli, and E. Feron. Rotorcraft motion planning for agile maneuvering. In *Proc. 58th Annual Forum of the American Helicopter Society*, Montreal, Canada, June 2002.

[94] B. Mettler, M. Valenti, T. Schouwenaars, Y. Kuwata, J. How, J. Paunicka, and E. Feron. Autonomous UAV guidance build-up: Flight-test demonstration and evaluation plan. In *Proc. AIAA Guidance, Navigation, and Control Conference*, Austin, TX, August 2003. AIAA Paper 2003-5744.

[95] M.B. Milam, K. Mushambi, and R.M. Murray. A new computational approach to real-time trajectory generation for constrained mechanical systems. In *Proc. 39th IEEE Conference on Decision and Control*, pages 845–851, Sydney, Australia, December 2000.

[96] I. Mitchell, A. Bayen, and C. Tomlin. A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on Automatic Control*, 50(7):947–957, July 2005.

[97] I. Mitchell and C. Tomlin. Level set methods for computation in hybrid systems. In B. Krogh and N. Lynch, editors, *Hybrid Systems: Computation and Control*, volume 1790 of *Lecture Notes in Computer Science (LNCS)*, pages 310–323. Springer Verlag, 2000.

[98] M. Morari and J. Lee. Model predictive control: Past, present and future. *Computers and Chemical Engineering*, 23(4):667–682, May 1999.

[99] J. De Mot and E. Feron. Spatial distribution of two-agent clusters for efficient navigation. In *Proc. 42nd IEEE Conference on Decision and Control*, Maui, HI, December 2003.

[100] H.G. Nguyen, N. Pezeshkian, M. Raymond, A. Gupta, and J.M. Spector. Autonomous communication relays for tactical robots. In *Proc. 11th Int. Conference on Advanced Robotics*, Coimbra, Portugal, June 2003.

[101] Office of the Under Secretary of Defense for Acquisition, Technology and Logistics. Unmanned aerial vehicles roadmap 2002-2027, December 2002. Report Number: A809414.

[102] M. Oishi, C. Tomlin, V. Gopal, and D. Godbole. Addressing multiobjective control: Safety and performance through constrained optimization. In M.D. Di Benedetto and A. Sangiovanni-Vincentelli, editors, *Hybrid Systems: Computation and Control*, volume 2034 of *Lecture Notes in Computer Science (LNCS)*, pages 459–472. Springer Verlag, 2001.

[103] L. Pallottino, E. Feron, and A. Bicchi. Conflict resolution problems for air traffic management systems solved with mixed integer programming. *IEEE Transactions on Intelligent Transportation Systems*, 3(1):3–11, 2002.

[104] G.J. Pappas, G. Lafferriere, and S. Sastry. Hierarchically consistent control systems. *IEEE Transactions on Automatic Control*, 45(6):1144–1160, June 2000.

[105] G.J. Pappas and S. Simic. Consistent hierarchies of nonlinear abstractions. In *Proc. 39th IEEE Conference on Decision and Control*, pages 4379–4384, December 2000.

[106] J. Paunicka, B. Mendel, and D. Corman. The OCP - an open middleware solution for embedded systems. In *Proc. 2001 American Control Conference*, pages 3445–3450, Arlington, VA, June 2001.

[107] M. Powers and T. Balch. Value-based communication preservation for mobile robots. In *Proc. 7th International Symposium on Distributed Autonomous Robotic Systems*, Toulouse, France, June 2004.

[108] A. Raghunathan, V. Gopal, D. Subramanian, L. Biegler, and T. Samad. Dynamic optimization strategies for three-dimensional conflict resolution of multiple aircraft. *AIAA Journal of Guidance, Control and Dynamics*, 27(4):586–594, 2004.

[109] J.B. Rawlings. Tutorial overview of model predictive control. *IEEE Control Systems Magazine*, 20(3):38–52, June 2000.

[110] J. Reif. Complexity of the mover's problem and generalizations. In *Proc. 20th IEEE Symposium on Foundations of Computer Science*, pages 224–241, 1979.

[111] J.H. Reif and M. Sharir. Motion planning in the presence of moving obstacles. In *Proc. 25th IEEE Symposium on Foundations of Computer Science*, pages 144–154, December 1985.

[112] A. Richards. *Robust Constrained Model Predictive Control*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, December 2004.

[113] A. Richards, J. Bellingham, M. Tillerson, and J. How. Coordination and control of multiple UAVs. In *Proc. AIAA Guidance, Navigation, and Control Conference*, Monterey, CA, August 2002.

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

[115] A. Richards and J. How. Model predictive control of vehicle maneuvers with guaranteed completion time and robust feasibility. In *Proc. 2003 American Control Conference*, Denver, CO, June 2003.

[116] A. Richards, Y. Kuwata, and J. How. Experimental demonstrations of real-time MILP control. In *Proc. AIAA Guidance, Navigation, and Control Conference*, Austin, TX, August 2003.

[117] A. Richards, T. Schouwenaars, J. How, and E. Feron. Spacecraft trajectory planning with avoidance constraints using mixed-integer linear programming. *AIAA Journal of Guidance, Control and Dynamics*, 25(4):755–764, 2002.

[118] E. Roche. Parsing with finite-state transducers. In E. Roche and Y. Schabes, editors, *Finite-State Language Processing*. MIT Press, Cambridge, MA, 1997.

[119] A.B. Roger and C.R. McInnes. Safety constrained free-flyer path planning at the international space station. *AIAA Journal of Guidance, Control, and Dynamics*, 23(6):971–979, 2000.

[120] J.A. Rossiter, B. Kouvaritakis, and J.R. Gossner. Guaranteeing feasibility in constrained stable generalized predictive control. *IEE Proc.-Control Theory Applications*, 143:463–469, 2001.

[121] E. Royer and C-K. Toh. A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications*, 6(2):46–55, 1999.

[122] T. Samad and G. Balas. *Software-Enabled Control: Information Technology for Dynamical Systems*. Wiley-IEEE Press, Hoboken, NJ, 2003.

[123] D.P. Scharf, F.Y. Hadaegh, and S.R. Ploen. A survey of spacecraft formation flying guidance and control (Part I): Guidance. In *Proc. 2003 American Control Conference*, pages 1733– 1739, June 2003.

[124] D.P. Scharf, F.Y. Hadaegh, and S.R. Ploen. A survey of spacecraft formation flying guidance and control (Part II): Control. In *Proc. 2004 American Control Conference*, pages 2976– 2985, June 2004.

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

[126] T. Schouwenaars, E. Feron, and J. How. Hybrid model for receding horizon guidance of agile autonomous rotorcraft. In *Proc. 16th IFAC Symposium on Automatic Control in Aerospace*, St. Petersburg, Russia, June 2004.

[127] T. Schouwenaars, B. De Moor, E. Feron, and J. How. Mixed integer programming for multi-vehicle path planning. In *Proc. 2001 European Control Conference*, pages 2603–2608, Porto, Portugal, September 2001.

[128] T. Schouwenaars, B. Mettler, E. Feron, and J. How. Robust motion planning using a maneuver automaton with built-in uncertainties. In *Proc. 2003 American Control Conference*, Denver, CO, June 2003.

[129] T. Schouwenaars, M. Valenti, E. Feron, and J. How. Implementation and flight test results of MILP-based UAV guidance. In *Proc. 2005 IEEE Aerospace Conference*, Big Sky, MT, March 2005.

[130] C. Schumacher, P.R. Chandler, and S. Rasmussen. Task allocation for wide area search munitions via network flow optimization. In *Proc. 2002 American Control Conference*, Anchorage, AK, May 2002.

[131] J.T. Schwartz and M. Sharir. On the piano movers' problem: I. *Communications on Pure and Applied Mathematics*, 36:345–398, 1983.

[132] J.T. Schwartz and M. Sharir. A survey of motion planning and related geometric algorithms. *Artificial Intelligence*, 37(1-3):157–169, 1988.

[133] J.T. Schwartz and M. Sharir. *Handbook of Theoretical Computer Science (vol. A): Algorithms and Complexity*, chapter Algorithmic Motion Planning in Robotics, pages 391–430. MIT Press, Cambridge, MA, 1990.

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

[135] P.O.M. Scokaert, D.Q. Mayne, and J.B. Rawlings. Suboptimal model predictive control (feasibility implies stability). *IEEE Transactions on Automatic Control*, 44(3):648–654, March 1999.

[136] P.O.M. Scokaert, J.B. Rawlings, and E.S. Meadows. Discrete-time stability with perturbations: Application to model predictive control. *Automatica*, 33(3):463–470, 1997.

[137] D.H. Shim, H.J. Kim, and S. Sastry. Decentralized nonlinear model predictive control of multiple flying robots in dynamic environments. In *Proc. 43rd IEEE Conference on Decision and Control*, Maui, HI, December 2003.

[138] K. Shin and N. McKay. A dynamic programming approach to trajectory planning of robotic manipulators. *IEEE Transactions on Automatic Control*, 31(6):491–500, June 1986.

[139] L. Singh and J. Fuller. Trajectory generation for a UAV in urban terrain, using nonlinear MPC. In *Proc. 2001 American Control Conference*, pages 2301–2308, Arlington, VA, June 2001.

[140] J-J.E. Slotine and W. Li. *Applied Nonlinear Control*. Prentice Hall, Upper Saddle River, NJ, 1991.

[141] D. Spanos and R. Murray. Motion planning with wireless network constraints. In *Proc. 2005 American Control Conference*, Portland, OR, June 2005.

[142] K. Sprague, V. Gavrilets, I. Martinos, D. Dugail, B. Mettler, and E. Feron. Design and applications of an avionics system for a miniature acrobatic helicopter. In *Proc. 20th Digital Avionics Systems (DASC)*, Daytona Beach, FL, October 2001.

[143] J. Sweeney, T.J. Brunette, Y. Yang, and R. Grupen. Coordinated teams of reactive mobile platforms. In *Proc. 2002 IEEE Int. Conference on Robotics and Automation*, Washington, D.C., May 2002.

[144] H.A. Taha. *Operations Research, An Introduction*. Macmillan Publishing Company, New York, 4th edition, 1987.

[145] C-K. Toh. *Ad Hoc Mobile Wireless Networks: Protocols and Systems*. Prentice Hall Publishers, 2001.

[146] C. Tomlin, I. Mitchell, and R. Ghosh. Safety verification of conflict resolution maneuvers. *IEEE Transactions on Intelligent Transportation Systems*, 2(2), 2001.

[147] C. Tomlin, G.J. Pappas, and S. Sastry. Conflict resolution for air traffic management: a study in multi-agent hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):509–521, 1998.

[148] P. Ulam and R.C. Arkin. When good comms go bad: Communications recovery for multi-robot teams. In *Proc. 2004 IEEE International Conference on Robotics and Automation*, pages 3727–3734, New Orleans, LA, April 2004.

[149] M. Valenti, T. Schouwenaars, Y. Kuwata, E. Feron, J. How, and J. Paunicka. Implementation of a manned vehicle-UAV mission system. In *Proc. AIAA Guidance, Navigation, and Control Conference*, Providence, RI, August 2004.

[150] A. Wagner and R. Arkin. Multi-robot communication-sensitive reconnaissance. In *Proc. 2004 IEEE International Conference on Robotics and Automation*, pages 4674–4681, New Orleans, LA, April 2004.

[151] C.W. Warren. Multiple robot path coordination using artificial potential fields. In *Proc. 1990 IEEE Int. Conference on Robotics and Automation*, pages 500–505, Cincinnati, OH, May 1990.

[152] W.L. Winston. *Operations Research, Algorithms and Applications*. Duxbury Press, Belmont, CA, 3th edition, 1994.

[153] J. Wohletz, D. Castanon, and M. Curry. Closed-loop control for joint air operations. In *Proc. 2001 American Control Conference*, pages 4699–4704, Arlington, VA, June 2001.