

ABSTRACT

Title of dissertation: MOTION PLANNING AND CONTROLS
WITH SAFETY AND TEMPORAL
CONSTRAINTS

Yuchen Zhou, Doctor of Philosophy, 2016

Dissertation directed by: Professor John S. Baras
Department of Electrical & Computer Engineering

Motion planning, or trajectory planning, commonly refers to a process of converting high-level task specifications into low-level control commands that can be executed on the system of interest. For different applications, the system will be different. It can be an autonomous vehicle, an Unmanned Aerial Vehicle(UAV), a humanoid robot, or an industrial robotic arm. As human machine interaction is essential in many of these systems, safety is fundamental and crucial. Many of the applications also involve performing a task in an optimal manner within a given time constraint. Therefore, in this thesis, we focus on two aspects of the motion planning problem. One is the verification and synthesis of the safe controls for autonomous ground and air vehicles in collision avoidance scenarios. The other part focuses on the high-level planning for the autonomous vehicles with the timed temporal constraints.

In the first aspect of our work, we first propose a verification method to prove the safety and robustness of a path planner and the path following controls based on

reachable sets. We demonstrate the method on quadrotor and automobile applications. Secondly, we propose a reachable set based collision avoidance algorithm for UAVs. Instead of the traditional approaches of collision avoidance between trajectories, we propose a collision avoidance scheme based on reachable sets and tubes. We then formulate the problem as a convex optimization problem seeking control set design for the aircraft to avoid collision. We apply our approach to collision avoidance scenarios of quadrotors and fixed-wing aircraft.

In the second aspect of our work, we address the high level planning problems with timed temporal logic constraints. Firstly, we present an optimization based method for path planning of a mobile robot subject to timed temporal constraints, in a dynamic environment. Temporal logic (TL) can address very complex task specifications such as safety, coverage, motion sequencing etc. We use metric temporal logic (MTL) to encode the task specifications with timing constraints. We then translate the MTL formulae into mixed integer linear constraints and solve the associated optimization problem using a mixed integer linear program solver. We have applied our approach on several case studies in complex dynamical environments subjected to timed temporal specifications. Secondly, we also present a timed automaton based method for planning under the given timed temporal logic specifications. We use metric interval temporal logic (MITL), a member of the MTL family, to represent the task specification, and provide a constructive way to generate a timed automaton and methods to look for accepting runs on the automaton to find an optimal motion (or path) sequence for the robot to complete the task.

MOTION PLANNING AND CONTROLS WITH
SAFETY AND TEMPORAL CONSTRAINTS

by

Yuchen Zhou

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2016

Advisory Committee:
Professor John S. Baras, Chair/Advisor
Professor André L Tits
Professor Gilmer L. Blankenship
Professor Nuno Martins
Professor Yiannis Aloimonos

© Copyright by
Yuchen Zhou
2016

Acknowledgments

I owe my gratitude to all the people who have made this thesis possible and because of whom my graduate experience has been one that I will cherish forever.

First and foremost I'd like to thank my advisor, Professor John S Baras for giving me an invaluable opportunity to work on challenging and extremely interesting projects over the past five years. He has always made himself available for help and advice even when he is really busy. It has been a pleasure to work with and learn from such an knowledgeable expert in so many different fields.

My colleagues at the System Engineering Integration Laboratory (SEIL) and Autonomy Robotics and Cognition Lab (ARC) have enriched my graduate life in many ways, and I greatly appreciate your helps and supports.

I owe my deepest thanks to my family, my mother and father who have always stood by me and guided me through my career. I owe my great thanks to my girlfriend, Junyi Shen, who have encouraged me and supported me in difficult time of graduate life. I cannot express in words how much the gratitude I owe them.

I would like to acknowledge financial support from the Air Force Office of Scientific Research (AFOSR), National Science Foundation (NSF) and National Institute of Standards and Technology (NIST) for all the projects discussed herein.

Table of Contents

List of Figures	v
1 Introduction	1
1.1 Motion Planning with Safety Constraints	3
1.2 Temporal Logic	6
2 Reachable Sets for Safety Verification	9
2.1 Overview	9
2.2 Reachable Set Computation	10
2.2.1 Reachable Set Computation for Linear Dynamics	12
2.2.2 Reachability Analysis for Hybrid System with Linear Dynamics	15
2.2.3 Reachability Analysis of Nonlinear Dynamics using Linearization	17
2.3 Reachability Analysis for Quadrotor Dynamics	19
2.3.1 Quadrotor Dynamics	19
2.3.2 Results and Analysis	21
2.4 Collision Avoidance Maneuver Safety Verification for Autonomous Vehicle	23
2.4.1 System Overview	24
2.4.2 Reachability for Physical System	26
2.4.3 Results and Analysis	28
2.5 Conclusion	30
3 Control Synthesis for Collision Avoidance Using Reachable Set	31
3.1 Overview	31
3.2 Reachable Set	34
3.3 Collision Avoidance Between Two Agents Using Reachability Analysis	37
3.3.1 Constant Control Set Design for Collaborative Collision Avoidance	37
3.3.2 Time Varying Control Tube Design for Collision Avoidance	44
3.4 Simulations and Results	49

3.4.1	Quadrotor Model	49
3.4.2	Fix-wing Dynamics Model	51
3.4.3	Reachable Sets and Constrained Control Sets for Constant Control Set Method	53
3.4.3.1	Quadrotor Example	53
3.4.3.2	Fix-wing Aircraft Example	58
3.4.4	Reachable Sets and Constrained Control Sets for the Time Varying Control Tube Method	61
3.5	Conclusion	64
4	Optimization based Temporal Logic Planning	65
4.1	Overview	65
4.2	Preliminaries	68
4.2.1	Metric Temporal Logic (MTL)	69
4.3	Problem Formulation and Solution	71
4.3.1	Linearized Dynamics of the Robot	72
4.3.1.1	Car-Like Model	72
4.3.2	Mixed Integer Linear Constraints	73
4.4	Case Study and Discussion	76
4.5	Conclusion	82
5	Timed Automata Based Motion Planning	85
5.1	Overview	85
5.2	Preliminaries	88
5.2.1	Metric Interval Temporal Logic (MITL)	88
5.2.2	MITL and timed automata based approach	90
5.3	MITL for motion planning	92
5.3.1	MITL to Timed Automata Transformation	92
5.3.2	Path Synthesis using UPPAAL	97
5.4	Case Study and Discussion	99
5.5	Continuous Trajectory generation	104
5.6	Conclusion	108
6	Conclusion	110
A	112
Bibliography	112
Bibliography	113

List of Figures

2.1	Planned trajectory	22
2.2	Reachable set computation of the two UAVs	23
2.3	Block diagram of the collision avoidance maneuver.	24
2.4	Reachable set of linear hybrid system model	27
2.5	Sensor noise influence on reachable sets	28
3.1	The initial reachable sets of both aircraft	54
3.2	The initial reachable tubes of of both aircraft	55
3.3	The initial control set and the constrained control sets for $k = 1$	56
3.4	The reachable tubes for aircraft A and B with $k = 1$	56
3.5	The initial control set and the constrained control sets for $k = 0.9$	57
3.6	The reachable tubes for aircraft A and B with $k = 0.9$	57
3.7	The initial reachable tubes of both fixed-wing aircraft	59
3.8	The initial control sets of both fixed-wing aircraft	60
3.9	The reachable tubes of both fixed-wing aircraft	61
3.10	The initial reachable tubes of both aircraft using predicted intruder tube	62
3.11	The reachable tubes of both aircraft using time varying control tube	63
4.1	Workspace setup of the first test case	77
4.2	Time-state-space representation of the environment and resulting trajectory with temporal constraints ϕ_1	78
4.3	Time-state-space representation of the environment and resulting trajectory with temporal constraints ϕ_2	79
4.4	Time-state-space representation of the environment and resulting trajectory with temporal constraints ϕ_1 for the car model	80
4.5	Time-state-space representation of the environment and resulting trajectory for the second setup	81
4.6	Time-state-space representation of the environment and resulting trajectory for the quadrotor with temporal constraints ϕ_3	82
4.7	Time-state-space representation of the environment and resulting trajectory for the quadrotor with temporal constraints ϕ_4	83

4.8	Time-state-space representation of the environment and resulting trajectory for the car model with temporal constraints ϕ_3	84
5.1	Timed Automata based on cell decomposition and robot dynamics . .	88
5.2	Logic tree representation of ϕ	92
5.3	The timed automaton for $p\mathcal{A}q$	94
5.4	The timed automaton for the generator part of $\diamond_I a$	95
5.5	The timed automaton for the checker part of $\diamond_I a$	96
5.6	Result timed automaton in UPPAAL of ϕ_1	100
5.7	Timed automata in UPPAAL corresponding to the generator of ϕ_2 . .	101
5.8	Timed automata in UPPAAL corresponding to the checker of ϕ_2 . . .	102
5.9	Workspace and the continuous trajectory for the specification ϕ_1 . . .	105

Chapter 1: Introduction

Motion planning, or trajectory planning, commonly refers to a process of converting high-level task specifications into low-level control commands that can be executed on the plant of interests. [1] For different applications, the plant will be different. It can be an autonomous vehicle, an Unmanned Aerial Vehicle(UAV), a humanoid robot or a industrial robotic arm. In many applications involving human machine interactions, safety and temporal constraints in the specifications are fundamental and crucial.

Recently, various automotive companies developed driver assistant systems using motion planning techniques. These systems commonly include either active or passive features interacting with driver. Common functions include lane departure warning, lane centering, lane changing, adaptive cruise controls and parking assist, which dramatically increase the autonomy of the vehicles [2]. General Motors (GM) developed the Super Cruise feature which allows hand-free highway driving with traffic [3]. Tesla Motors deployed their Autopilot feature which allows autonomous lane changing, lane centering and autonomous parking. [4]. Google's fully autonomous vehicles emphasize on urban environment navigation, such as following traffic lights and stop signs, avoiding pedestrians and other cars on the road. Most of these

semi-autonomous or autonomous functions are safety critical and it is essential to formally prove their safety properties.

Similarly in aviation, safety and temporal constraints play a key role in the motion planning process. UAVs have been deployed for agriculture research and management, surveillance and sensor coverage for threat detection and disaster search and rescue operations. If in the future, autonomous UAVs are allowed to be operated autonomously in the same airspace as the human piloted aircraft, it is absolutely crucial to keep the UAVs a safe distance away from the human piloted plane to avoid catastrophic accidents. Additionally, these applications require the tasks to be finished in an optimal manner with specific timing constraints. Surveillance and sensor coverage, for example, requires the areas to be visited repeatedly and sequentially. These specifications commonly have timed duration associated, since data collection requires time and moving obstacles in the workspace block spaces for some amounts of time. Therefore, it is essential to have timed temporal logic specifications which captures temporal ordering and duration constraints. To conclude, in aviation, motion planner should be developed while considering both safety and temporal constraints.

In this thesis, we focus on two aspects of the motion planning problem. One is the verification and synthesis of the safe controls for the autonomous ground and air vehicles in collision avoidance scenarios. The other part focuses on the high-level planning for the autonomous vehicles with the timed temporal constraints. The following sections give overviews of the scope and motivation of these two parts.

1.1 Motion Planning with Safety Constraints

In robotics, kinodynamic motion planning, refers to a particular type of motion planning, that requires the trajectory generated to be kinematically feasible and satisfies the dynamic equations of the robot. Exact trajectory is commonly computational infeasible, and costly to implement in real-time. [5] Because of difficulties of planning a exact dynamic feasible trajectory in real-time, commonly, a piecewise polynomial function (spline) [6], or other types of piecewise smooth functions [7] are used as a reference trajectory for the controller. Because switching between reference trajectories can cause unmodeled transient behaviors, it is essential to provide safety guarantees in particular for safety critical applications, such as collision avoidance maneuvers for automobile and UAVs. Therefore, we focus on safety requirements which are separation requirements in space and time.

Consider the following collision avoidance scenario, where individual agents have been assigned different goals independently, and their paths may cross if they go directly towards their goals. Many motion planning algorithms have been proposed in the robotics areas to solve this problem. An artificial potential function based algorithm was proposed in [8–10] to produce control policies for robots to navigate towards their goals while avoiding each others and the obstacles in the workspace. In [11–13], the authors proposed a decentralized collision avoidance algorithm based on the heading of the vehicles and collision cones constructed from relative velocities. In [7], the authors proposed a sampling based method to generate a kinematically feasible path that is collision. However, the research works

mentioned above focus on designing only a single path or trajectory for an individual vehicle, so that they are separated by at least the required distance. Violations of the separation requirements while following the trajectory can be caused by modeling errors, sensor noises, discretizations, linearizations of the nonlinear models and robustness of the trajectory following algorithms.

We propose to improve the safety of the motion plan in two different perspectives. One is the verification of the safety requirement of the reference trajectories and trajectory following controls, the other is the synthesis of the controls to fulfill the safe margins. Both perspectives require models of deviations from the reference trajectory due to uncertainties and disturbances in the system.

One way to capture deviations in system modeling is to use the reachable set, which is a set of states that is reachable from the initial set given the models of uncertainties and disturbances. The safety requirements such as a minimum separation between individual agent becomes a requirement on the distances between the reachable sets over time. In computational mathematical terms, the reachable set computation can be carried out through different over-approximations or under-approximations. The set can be captured using intervals, zonotopes, ellipsoids, polytopes, support functions, level sets of Hamiltonian.

For general nonlinear dynamics, in [14, 15] the authors proposed to formulate the reachability problem into a game theoretic optimization problem and represented the reachable set as a level set computed by solving Hamilton Jacobi Issac (HJI) equations. Due to the computational complexity of solving fixed boundary partial differential equations, the reachable sets computed are restricted to dimension lower

than 3. This restricts the problem to kinematics only, where forces and accelerations are not modeled. In most of the physical systems in motion planning problems, because actuators can saturate due to physical limitation or dynamics, it is crucial to consider the bounds on the control inputs. Other computational efficient algorithms commonly rely on the convexity of the sets and linear dynamics. Reachable sets can be over-approximated using specific convex covering sets, such as ellipsoids [16,17], convex sets described by support function [18], and polytopes [19–22]. These methods can handle cases of high dimensional systems resulting from quadrotor or manipulator dynamics which have high dimension of state space. Although the underlying computation is based on linear techniques, the system dynamics can be nonlinear [23], or hybrid [24].

For the verification problem, we want to prove that given the reference trajectory and the trajectory following controls, the system is safe and robust to modeled disturbance in the system. The system is safe if it satisfies the separation requirement at all times for all possible modeled disturbances and inputs. In the UAV collision avoidance example, we proposed a method to prove the safety of a planned UAV trajectory and an expected human piloted trajectory using the reachable sets. For collision avoidance maneuver for automobile, we demonstrated our method to validate lateral separation requirement with the front vehicle. In both applications, efficient computation for high dimensional systems is important. Furthermore, in these cases, the system is commonly operating in linear regions or near steady state. Therefore, we deployed methods based on linear and convex reachable set computation. We covered the details of the verification algorithms and demonstrated the

method for UAV and cars through simulations in Chap. 2.

From the control synthesis point of view, we proposed an update rule to restrict the control sets of the vehicles so that the reachable sets of all the vehicles on collision courses, are safe respect to any execution of the others. This provides guarantees for unknown control actions, within some negotiated control constraints, of the other vehicles, so that the host vehicle can avoid them under all possible disturbances and controls. We examined such method for both rotorcraft and fixed-wing dynamics in Chap. 3.

1.2 Temporal Logic

In recent years, there has been increasing interest in using temporal logic, such as Linear Temporal Logic (LTL), Computation Tree Logic (CTL) and regular expressions mission planning at the high level [25–30]. Temporal logics [31–33] have provided a compact mathematical formulation for specifying complex motion specifications, motion sequencing and timing behaviors etc. For example, high-level mission specifications such as “Visit region R1, then R2, and then R3, infinitely often. Never enter R5 unless coming directly from R4.” can be specified in LTL. Previous approaches mainly focus on the usage of LTL, which can specify tasks such as visiting goals, periodically surveying areas, staying stable and safe. Historically LTL was originated for model checking, validation and verification in the software community [34]. The availability of tools such as SPIN [35], NuSMV [36] made it easier for us to check if a given LTL specifications can be met by creating a suitable

automaton and looking for a feasible path on that automaton. However, the construction of the automaton from a given temporal logic formula is based on the fact that there are no time constraints associated with the specification. That is also the main drawback of the LTL formulation, which is that it cannot specify time distance between tasks. Currently motion planning for robots is in such a stage where it is very crucial to incorporate time constraints since these constraints can arise from different aspects of the problem: dynamic environment, sequential processing, time optimality etc. In surveillance examples, a simple task may be to individually monitor multiple areas for at least x amount of time while satisfying the constraints of visiting the areas at least once every y amount of time. Planning with time bounded objectives is inherently hard due to the fact that every transition from one state to another in the associated automaton has to be carried out, by some controller, exactly in time from an initial configuration to the final configuration. Time bounded motion planning has been done in heuristic ways [37, 38]. We proposed two very different methods to solve the planning under these timed temporal constraints. One is transforming the problem into a mixed integer optimization problem, the other is generating an associated timed automaton that captured the timed temporal constraints similar to how a constructed Büchi Automaton can represent the same constraints in LTL.

In Chap. 4, we first define the temporal logic we used in this work, Metric Temporal Logic (MTL). It allows us to define tasks with timed temporal constraints. We then propose a method to transform the problem into a mixed integer linear program, and solved it through commercial solvers such as CPLEX.

In Chap. 5, we first review previous LTL automata-based approaches and commented on the difficulties for MTL. A modified version of the MTL, Metric Interval Temporal Logic (MITL), is used. Secondly, we propose an autonomous method to transform the MITL to a timed automaton based on a model checking method in computer science. We then analyze the timed automaton and find shortest time discrete plan in UPPAAL. UPPAAL is a timed automata analysis and verification tool. Lastly, the discrete plans generated from UPPAAL is converted to continuous trajectories through optimal controls.

Chapter 2: Reachable Sets for Safety Verification

2.1 Overview

As discussed in the previous chapter, reachable sets can be used to evaluate robustness and safety properties of motion planning and control algorithms. Commonly in practice of robotics, a reference trajectory is generated from motion planning, and a model predictive controller (MPC) [39,40] or an adaptive controller [41] are used to follow the resulting trajectory. Conventional MPC based methods have robustness and stability problems, and adaptive control based methods can cause unexpected transient behaviors before convergence of the controller. For safety critical systems, such as a collision avoidance module, it is essential to make sure that the overall system is safe under disturbances during the maneuver. Reachable sets can capture uncertainties and deviations from the nominal trajectory as well as transient behavior of the trajectory following controllers.

In this chapter, we will first provide fundamentals of the reachable sets including definition and computation using iterative methods. We then show how the algorithm can be used to verify robustness of the reference trajectory and the trajectory tracking algorithm in nonlinear and hybrid system settings. The methods are then demonstrated on two dynamic systems, a quadrotor and a car dynamics,

in a collision avoidance scenario. A part of this work is published in [42].

2.2 Reachable Set Computation

We consider the collision avoidance problem between vehicles whose dynamics are given by nonlinear models as (2.1).

$$\dot{x}(t) = f(t, x, u, v) \tag{2.1}$$

where $x(t) \in \mathcal{X}$, $x(0) \in \mathcal{X}_0 \subseteq \mathcal{X}$, $u(t) \in \mathcal{U}(t)$ for all t , $v(t) \in \mathcal{V}$ for all t . \mathcal{X}_0 is the initial state set, $\mathcal{U}(t)$ is the control set and \mathcal{V} is the disturbance set.

The reachable set of (2.1) (or forward reachable set) $\mathcal{R}[\vartheta] = \mathcal{R}(\vartheta, X_0)$, is the set of states that are reachable at time ϑ from a set of initial states X_0 and all possible controls and disturbances. Formally it is defined by the following,

Definition 2.2.1 (Reachable Set). *The reachable set $\mathcal{R}[\vartheta] = \mathcal{R}(\vartheta, t_0, X_0)$ of the system of (2.1) at time ϑ from a set of initial positions X_0 and time t_0 is the set of all points x for which there exists a trajectory $x(s, t_0, x_0, u(s), v(s))$, $x_0 \in X_0$, $u(t) \in \mathcal{U}(t)$, $v(t) \in \mathcal{V}$ for all t , that transfers the system from (t_0, x_0) to (ϑ, x) , $x = x(\vartheta)$.*

Similarly the reachable tube is the union of all reachable sets over a time interval.

Definition 2.2.2 (Reachable Tube). *The reachable tube $\mathcal{R}(\Theta, X_0) = \cup_{\vartheta \in \Theta} \mathcal{R}(\vartheta, t_0, X_0)$*

For large dimensional systems such as aircraft, effective reachable set computations using convex overapproximation include the convex sets described by support functions [18, 43], polytopes [22, 44] and zonotopes [23, 45–49]. We use the zonotopes

representation (Def. 2.2.3 below), because the computation of the zonotope over-approximations can be performed efficiently for linear systems, linearized nonlinear systems, and hybrid linear systems [23, 49].

Definition 2.2.3 (Zonotope). *A **zonotope** is a set of points in n -dimensional space constructed from vectors $v_i \in \mathbb{R}^n$ by taking the sum of $a_i v_i$, where a_i is a scalar between 0 and 1. Formally, the zonotope $\mathcal{Z}(v_1, v_2, \dots, v_N) = \left\{ z \in \mathbb{R}^n \mid z = \sum_{i=1}^N a_i v_i \right\}$, where $0 \leq a_i \leq 1$.*

The tool Continuous Reachability Analyzer (CORA), which implements zonotope-based reachability analysis, runs in the Matlab environment. It has been implemented with the capability of verifying the safety of highly nonlinear autonomous vehicles traversing a reference trajectory [45–47]. It is very appropriate for the motivating applications. Furthermore, the zonotope has been extended further to consider affects of time varying system [48], and introduce a better linearization scheme to reduce error introduced [23].

Reachable set computation for a nonlinear model directly is commonly impractical in collision avoidance due to slow computation time for a dynamical system [14]. Currently, there are two ways to capture the nonlinearity through linear methods. One method is linearizing the dynamics around a steady state and capturing the linearization error as part of bounded disturbance. [19] The other method is that we can convert the nonlinear dynamics to piecewise affine dynamics, which is especially useful to model saturation. This piecewise affine dynamics can be captured in a hybrid system with affine dynamics.

In the next few sections, computation of the reachable set through an iterative method will be summarized for linear, nonlinear and hybrid systems. It is based on algorithms proposed in [21, 43].

2.2.1 Reachable Set Computation for Linear Dynamics

Consider a simple linear time system described by the following,

$$\dot{x}(t) = A(t)x(t) + B(t)u(t) + v(t), \quad x(0) \in \mathcal{X}_0, \quad u(t) \in \mathcal{U}(t), \quad v(t) \in \mathcal{V} \quad (2.2)$$

where \mathcal{X}_0 is the initial state set, that satisfies $\mathcal{X}_0 \subset \mathcal{X} \subset \mathbb{R}^n$. \mathcal{X} is the state set. $\mathcal{U}(t) \subset \mathbb{R}^m$ is the control set at time t . $\mathcal{V} \subset \mathbb{R}^n$ is a bounded disturbance set that captures uncertainties in the system.

The solution of Eq. 2.2 for $x(0) = x_0$ is

$$x(t) = \Phi(t, 0)x_0 + \int_0^t \Phi(t, \tau)B(\tau)u(\tau)d\tau + \int_0^t \Phi(t, \tau)v(\tau)d\tau \quad (2.3)$$

where $\Phi(t, s)$ is the state transition matrix of system (2.2). Therefore, it satisfies

$$\frac{\partial}{\partial t}\Phi(t, s) = A(t)\Phi(t, s), \quad \Phi(s, s) = \mathbf{I}.$$

The solution (2.3) is a sum of three terms. One is due to the initial condition, one is due to the control and the last is related to the disturbances. Consider a trajectory tracking control law which is affine in state $x(t)$ based on MPC, i.e.

$$u(t) = -K(t)x(t) + h(\gamma(t), t)$$

where $K(t)$ and h are derived based on the cost function in MPC and the reference trajectory $\gamma(t) : \mathbb{R}_+ \mapsto \mathcal{X}$. The closed loop system will be a linear dynamics related

to x and w .

$$\dot{x}(t) = A_c(t)x(t) + \underbrace{v(t) + B(t)h(\gamma(t), t)}_{w(t) \in \mathcal{W}(t)}, \quad x(0) \in \mathcal{X}_0, \quad v(t) \in \mathcal{V} \quad (2.4)$$

where $A_c(t) = A(t) - B(t)K(t)$ is the matrix of the closed loop system. $\mathcal{W}(t)$ can be considered as a disturbance set with a known offset due to $h(\gamma(t), t)$.

The reachable set of the system (2.4) given the initial set \mathcal{X}_0 and the disturbance set $\mathcal{W}(t)$, can be captured as Minkowski sum of two sets (Def. 2.2.4 below), one from the initial state set \mathcal{X}_0 , one from the disturbance set $\mathcal{W}(t)$. Denote $V = \mathcal{R}(\delta, \{0\})$

$$\mathcal{R}(\delta, \mathcal{X}_0) = \Phi_c(\delta, 0)\mathcal{X}_0 \oplus \mathcal{R}(\delta, \{0\}).$$

$\mathcal{R}(\delta, \mathcal{X}_0)$ is the reachable set of the system (2.4) at time δ based on Def. 2.2.1.

$\mathcal{R}(\delta, \{0\})$ is the reachable set contributed due to disturbance set only, i.e. $x_0 = 0$.

$\Phi_c(t, s)$ is the transition matrix for the closed-loop system.

Definition 2.2.4. *The **Minkowski Sum** of two sets $A \in \mathbb{R}^n$ and $B \in \mathbb{R}^n$ is a set formed by additions of vectors in the two sets, i.e.*

$$A \oplus B = \{a + b \mid a \in A, b \in B\}$$

This suggests a discretized iterative computation of the reachability tube as the union of Ω_i , where Ω_i is defined by the following induction,

$$\Omega_{i+1} \leftarrow \Phi_i \Omega_i \oplus V.$$

$\Phi_i = \Phi_c((i+1)\delta, i\delta)$, $V = \mathcal{R}(\delta, \{0\})$, and δ is the sampling rate of the discretization.

The \oplus operator is the minkowski sum operator defined in Def. 2.2.4. Then the

induction is initiated by $\Omega_0 = \mathcal{R}([0, \delta], \mathcal{X}_0)$, with W . To reduce computational complexity, both V and Ω_0 are overapproximations.

The objective of finding the reachable set during the interval $[0, T]$ can then be computed using Algorithm 1

Algorithm 1 Reachable Set for Linear System

Input: Initial set \mathcal{X}_0 , horizon T , sampling step N .

Output: Reachable set $\mathcal{R}([0, T], \mathcal{X}_0)$

- 1: $\delta \leftarrow T/N$
 - 2: $r_\delta \leftarrow \sup_{w(t) \in \mathcal{W}(t)} \int_0^\delta \Phi_c(t, \tau) w(\tau) d\tau$
 - 3: $V \leftarrow \mathcal{B}(r_\delta)$ ▷ $\mathcal{R}(\delta, \{0\})$ overapproximated by V
 - 4: $s_\delta \leftarrow \sup_{x \in \mathcal{X}_0} (\Phi_c(\delta, 0) - I - A_c(\delta))x$
 - 5: $\Omega_0 \leftarrow CH(\mathcal{X}_0 \cup \Phi_0 \mathcal{X}_0) \oplus \mathcal{B}(s_\delta + r_\delta)$ ▷ $\mathcal{R}([0, \delta], \mathcal{X}_0)$ overapproximated by Ω_0
 - 6: **for** i from 0 to $N - 2$ **do**
 - 7: $\Omega_{i+1} \leftarrow \Phi \Omega_i \oplus V$ ▷ Computationally expensive
 - 8: **end for**
 - 9: **return** $\bigcup_{i \in \{0, \dots, N-1\}} \Omega_i$
-

$\mathcal{B}(r)$ is the ball with radius r , and $CH(\mathcal{Y})$ is the convex hull of the set \mathcal{Y} .

Explicit expressions of the approximations in line 3 and 5 using matrix norm was proposed in [49] and [48] for time invariant and time varying system. In both cases, the approximated reachability set converges to the real one as $\delta \rightarrow 0$.

The explicit form of line 7 is,

$$\Omega_i = \left(\prod_{j=0}^i \Phi_j \right) \Omega_0 \oplus \left[\bigoplus_{j=0}^{i-1} \left(\prod_{k=0}^j \Phi_k \right) V \right] \quad (2.5)$$

Because the computation of Ω_i has an increasing computational complexity for every iteration in line 7, an iteratively overapproximation by (2.6) is proposed

in [48, 49].

$$\left. \begin{aligned} \mathcal{A}_i &\leftarrow \Phi_i \mathcal{A}_{i-1} \\ \mathcal{S}_i &\leftarrow \mathcal{S}_{i-1} \oplus V_{i-1} \\ V_i &\leftarrow \Phi_{i-1} V_{i-1} \\ \Omega_i &\leftarrow \mathcal{A}_i \oplus \mathcal{S}_i \end{aligned} \right\} i = 1, 2, \dots, N \quad (2.6)$$

where the induction is initialized by, $\mathcal{A}_0 \leftarrow \Omega_0, V_0 \leftarrow V, \mathcal{S}_0 \leftarrow \{0\}$.

2.2.2 Reachability Analysis for Hybrid System with Linear Dynamics

The hybrid model of the overall system is formally defined as the following based on [50], [43].

Definition 2.2.5 (Hybrid Automata). *A hybrid automaton H is a tuple $H = (Q, X, Flow, \mathcal{I}, Trans, S_0)$ defined by the following components:*

- Q : a finite set of discrete locations q_i , which are vertices of a graph (V, E) whose edges are discrete transition defined by $Trans$.
- X : a finite number of continuous state variables $[x_1, \dots, x_{\|X\|}] \in R^{\|X\|}$ whose dynamics is defined by $Flow$ of every location q_i .
- $Flow$: a set of predicates on the derivatives of continuous variable, more specifically, $Flow(q)$ specifies the set for allowed $X \times \dot{X}$. The particular flow condition we are used here are general nonlinear systems in the form of $\dot{x} = f(x, w)$, for $w \in \mathcal{W}$. If the underlying dynamics at each location is linear then convex \mathcal{W} is assumed.

- \mathcal{I} : invariant set is a set of predicates on the continuous variables X for each location. If the underlying dynamics is linear then convex invariant constraint is assumed.
- Trans : is defined by a set of transition $E \subseteq Q \times Q$ and its associated jump predicate G on X and jump function μ . The transition from q to q' is taken, if there exists a transition from q to q' where its predicate G is satisfied, then next state $x' = \mu(x)$. Denote the set of state satisfying G as \mathcal{G} .
- S_0 : a set of initial state $Q_0 \times \mathcal{X}_0 \subset Q \times X$.

In the linear time invariant cases, where the flows of the hybrid automaton are linear time invariant in x , additional assumptions are the invariants and guard conditions are convex and the jump functions are affine in x and w . This section summarizes the approach in [43] of reachability analysis for linear system and extension to hybrid system.

To extend the previous framework to hybrid systems, one needs to add invariant conditions on x and guard and jump conditions on x .

To handle the invariants, i.e. the dynamics of the system in one location is,

$$\dot{x}(t) = A_l x(t) + w(t) \quad x(0) \in \mathcal{X}_0, w(t) \in \mathcal{W}(t), x(t) \in \mathcal{I}.$$

The reachable set should stay in the invariant \mathcal{I} , in other words, the reachable set Ω_i should also intersect the invariant set every iteration. This results in the modification of Eq. 2.5 to

$$\Omega_{i+1}^{\mathcal{I}} = (\Phi \Omega_i^{\mathcal{I}} \oplus V) \cap \mathcal{I}$$

The associated halfspace \mathcal{G} associated with the guard condition can then make intersection with the resulting reachable set. Let the intersection of the reachable set with the hyperplane happen between t_1 and t_2 , then the next step initial set can be formally described by the following,

$$X'_0 = \mu(\mathcal{R}([t_1, t_2], \mathcal{X}_i) \cap \mathcal{G}),$$

where μ is the associated jump function defined by the hybrid system.

2.2.3 Reachability Analysis of Nonlinear Dynamics using Linearization

The above sections summarize the work in reachability related to hybrid systems with linear dynamics and convex constraint set. Several researches have been done to extend the reachability framework for linear systems to nonlinear systems using linearization and hybridization techniques. Both techniques use abstraction to a differential inclusion of simpler dynamics such as hybrid systems with linear dynamics [51], or linearized around reference trajectories [45–47]. The latter outperforms partition-based methods because partition-based methods are commonly fixed. Abstraction using linearization is useful for dealing with very complex systems such as high-order car models [47], helicopters and high-order aircraft models. In general, the reachability analysis using abstraction has large advantage in computational efficiency over Hamilton Jacobean Integration (HJI) Equation based optimization algorithms in [14, 52–54].

Consider the general nonlinear system described by the following ODE and its

Taylor series expansion,

$$\begin{aligned}\dot{x}(t) &= f(x, w) \\ &= f(x^*, w^*) + \frac{\partial f}{\partial x} \Big|_{x=x^*} (x - x^*)(t) + \frac{\partial f}{\partial w} \Big|_{w=w^*} (w - w^*)(t) \\ &\quad + \text{higher order term,} \quad x(0) \in \mathcal{X}_0, w(t) \in \mathcal{W}(t),\end{aligned}$$

x^*, w^* can be computed based on nominal trajectory and nominal disturbance. Note that $w(t)$ in Eq. 2.4 also include the term $h(\gamma(t), t)$ related to the reference trajectory.

Furthermore, the nonlinear dynamics can be differential included by the linear dynamics, if the Lagrangian remainder terms of the Taylor series expansion is carefully computed. A reference trajectory $x_d(t)$ is added as a control input. Let $z = [x, w]^T$, then the differential inclusion can be formally described as,

$$\dot{z}(t) \in f(z^*, x_d) + \nabla_z f|_{z=z^*} (z - z^*)(t) + \frac{1}{2}(z - z^*)^T \nabla_z^2 f|_{z=\xi} (z - z^*), \quad (2.7)$$

where ξ takes value in the set $\{z + \alpha(z - z^*) | \alpha \in [0, 1]\}$.

In other words, the reachable sets of the linearized system, enlarged by the Lagrangian remainder terms, always include the reachable set of the nonlinear dynamics. Therefore, one can determine if the system is safe based on the overapproximated reachable set of the linearized dynamic. The overapproximation is also conservative in the sense that the overapproximation set is tight. As the discretization $\delta \rightarrow 0$, the overapproximated reachable set will converge to the real set.

2.3 Reachability Analysis for Quadrotor Dynamics

One of the important usages of reachability analysis is to improve safety of autonomous aircraft such as quadrotors in commercial airspace.

2.3.1 Quadrotor Dynamics

A common second order quadrotor model has a 12-dimensional state variable x and control input u based on thrust and rotational momentum. [55] The components of x are defined by the following,

$$x = (p_x, p_y, p_z, v_x, v_y, v_z, \phi, \theta, \psi, p, q, r),$$

where $\mathbf{p} = (p_x, p_y, p_z)$ is a position vector in the global coordinate frame, $\mathbf{v} = (v_x, v_y, v_z)$ is a velocity vector in the global coordinate frame, (ϕ, θ, ψ) are roll pitch yaw angle associated to attitude of the quadrotor, (p, q, r) is body frame rotational velocity. The control inputs are described by a four dimensional input variable $\mathbf{u} = (F, M_1, M_2, M_3)$, where F is the total thrust along the centerline of the body frame, and $\mathbf{M} = (M_1, M_2, M_3)$ is the rotational momentum. The dynamics of the quadrotor is commonly described by the following ODEs.

$$\dot{\mathbf{p}} = \mathbf{v},$$

$$\dot{\mathbf{v}} = g \mathbf{e}_3 - \frac{F}{m} R \mathbf{e}_3$$

$$\dot{\mathbf{R}} = R_w^{-1} \boldsymbol{\Omega}$$

$$\dot{\boldsymbol{\Omega}} = J^{-1}(\mathbf{M} - \boldsymbol{\Omega} \times J \boldsymbol{\Omega})$$

where J is the moment of inertial, R is the rotational matrix described by Z-X-Y Euler angle for description of roll pitch yaw, and R_w is the matrix transformation from derivatives of the roll pitch yaw to the body frame rotational velocities. [55, 56]. Cosine and Sine is abbreviated to “c” and “s” for short.

$$R = \begin{pmatrix} c\theta c\psi - s\theta s\phi s\psi & -c\phi s\psi & c\psi s\theta + c\theta s\phi s\psi \\ c\psi s\theta s\phi + c\theta s\psi & c\phi c\psi & -c\theta c\psi s\phi + s\theta s\psi \\ -c\phi s\theta & s\phi & c\theta c\phi \end{pmatrix}$$

$$R_w = \begin{pmatrix} c\theta & 0 & c\phi s\theta \\ 0 & 1 & s\phi \\ s\theta & 0 & c\phi c\theta \end{pmatrix}$$

A very nice property of this dynamics or helicopter dynamics is differential flatness. In other words, system states and control inputs can be written as algebraic functions of carefully selected flat outputs and their derivatives. For quadrotor dynamics in particular, it is proved in [55], that the flat outputs can be $y = (p_x, p_y, p_z, \psi)$. It also has been proved that a tracking controller defined by the following equations provide asymptotic stability to the reference trajectory y_d .

$$\mathbf{e}_p = \mathbf{p}_d - \mathbf{p}, \quad \mathbf{e}_v = \mathbf{v}_d - \mathbf{v}$$

$$\mathbf{F}_{\text{des}} = K_p \mathbf{e}_p + K_v \mathbf{e}_v + m g \mathbf{e}_3 + m \mathbf{a}_d$$

$$F = \mathbf{F}_{\text{des}} \cdot \mathbf{z}_B$$

$$\mathbf{e}_R = 1/2(-R_d^T R + R^T R_d)^\vee, \quad \mathbf{e}_w = w_d - w$$

$$\mathbf{M} = K_R \mathbf{e}_R + K_w \mathbf{e}_w$$

$$u = [F; \mathbf{M}] = \mathbf{f}(\mathbf{x}, \mathbf{x}_d) \quad (2.8)$$

In Eq. 2.8, \mathbf{x}_d is a reference trajectory to the system, defined by

$$\mathbf{x}_d = [\mathbf{p}_d, \mathbf{v}_d, \mathbf{a}_d, \phi_d, \theta_d, \psi_d, \boldsymbol{\Omega}_d].$$

This dynamical model and controller of the hardware platform can be easily adjusted using information from existing hardware databases to model other quadrotor platforms.

The overall dynamics of the quadrotor then become the same as Eq. 2.7. The associated input set \mathcal{U} here is not a control input set, but rather a disturbance set which captures the model uncertainties, sensor noise, and disturbances.

2.3.2 Results and Analysis

Computation using reachable set based methods has been demonstrated for real-time applications of autonomous vehicles with high degrees of nonlinearity [47, 48]. The outcome of the reachability analysis includes reachable sets of position and orientation of autonomous aircraft along with piloted aircraft that are inside the surveying workspace. When there are nearby friendly aircraft which are autonomously controlled by the same high level control system, the reachability set data are shared and safety is verified by checking that the reachable sets are not colliding in any instant of the time. The UAV operator will be able to obtain the reachable tubes of the autonomous aircraft in almost real-time and transmit location and trajectory data to Air Traffic Control (ATC) and nearby aircraft if desired. Human piloted aircraft always have the highest priority in our system, and autonomous

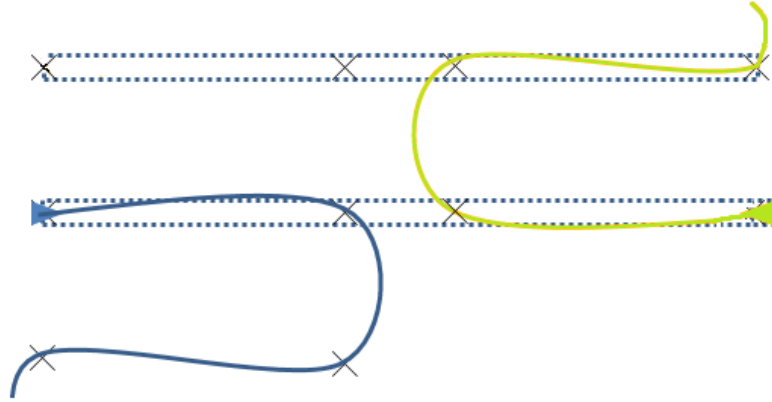


Figure 2.1: Planned trajectory. The blue and green arrows mark the initial positions of the UAVs. The crosses mark the waypoints for individual UAVs in a survey task. Possible information is marked by the dotted area. The corresponding curves are reference trajectories generated by the planner.

quadrotors, can reroute. If the reachable sets collide, the trajectory planned by the high level can be changed and the low level controller will be switched to an emergency avoidance state to use maximum power to move clear of the paths of piloted aircraft, this will be discussed in more detail in Chap. 3.

Consider the following scenario when two UAVs approach and avoid each other by negotiating different reference trajectories in the high level as Fig. 2.1. The safety verification in this case is to prove that the reachable set under disturbance is not large enough to collide the two sets. The result is shown in Fig. 2.2 with disturbance. The conclusion is that the reachable set has not enough separation due to the transient instability of trajectory tracking controller while switching between different reference trajectories. The computation time is 100x longer than the tested time horizon. This is because by introducing the time varying reference trajectory, the linearization and linearization error have to be computed at much higher frequency. One direction of future work is to improve efficiency for quadrotor

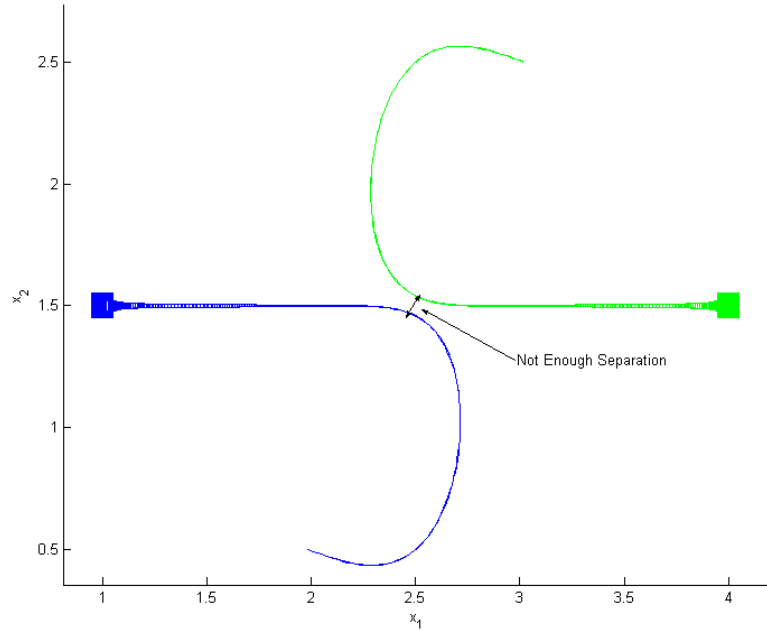


Figure 2.2: Reachable set computation of the two UAVs

reachable set computation, so that the safety verification can be performed real-time similar to the autonomous vehicle case.

2.4 Collision Avoidance Maneuver Safety Verification for Autonomous Vehicle

In this section, we demonstrated that the reachable set computation can be used for real-time verification of autonomous driving algorithms, such as collision avoidance maneuvers.

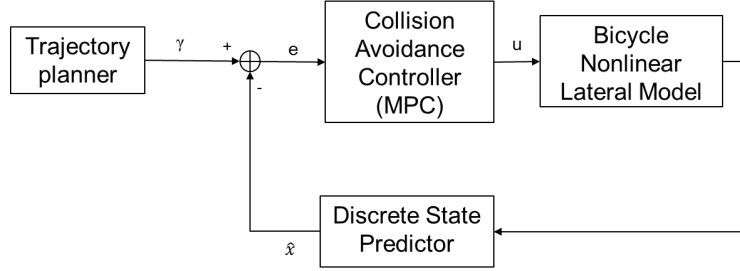


Figure 2.3: Block diagram of the collision avoidance maneuver.

2.4.1 System Overview

The most common collision avoidance maneuver model developed for semi-autonomous driving vehicle is composed of a path planner, a state predictor, the vehicle lateral dynamics, and a model predictive controller. [57–61] The interconnection is shown in Fig. 2.3. Based on lane camera updates of the road ahead of the vehicle, the path planner will generate a reference path for the vehicle to avoid an intimate collision by guiding the vehicle to left or right lane. The reference signal produced from the path planner γ captures the relative lateral distance and relative heading to the planned path. The state predictor estimates the future state of the vehicle state \hat{x} using current sensor measurements and discretized linear model of the dynamics. The error terms between the reference signal and the predicted state is fed through the model predictive controller to determine the control input to be used.

The lateral dynamics of the vehicle is a 4 dimensional model that consists of tracking error dynamics and classical forward steering bicycle dynamics. The states x are composed of relative lateral distance to the target path y , relative heading angle ψ , lateral speed v_y and yaw rate r . The controller input u is the front road wheel

angle δ . By using small angle approximation and linear tire model the following linearized state equations can be obtained.

$$\dot{x} = Ax + Bu + w$$

where

$$A = \begin{bmatrix} 0 & v_x & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\frac{C_f+C_r}{Mv_x} & -v_x - \frac{aC_f-bC_r}{Mv_x} \\ 0 & 0 & -\frac{aC_f-bC_r}{I_z v_x} & -\frac{a^2 C_f + b^2 C_r}{I_z v_x} \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \\ \frac{C_f}{M} \\ \frac{aC_f}{I_z} \end{bmatrix}, w = \begin{bmatrix} 0 \\ -\chi v_x \\ 0 \\ 0 \end{bmatrix}$$

Here v_x is the longitudinal speed, C_f and C_r are the cornering stiffness of the front and rear axles, respectively, a is the distance from the center of gravity of the vehicle to the front wheel axle, b is the distance from the center of gravity to the rear wheel axle, M is the total mass of the vehicle, I_z is the moment of inertia of the vehicle in the yaw direction, χ is the path curvature.

The linear bicycle model uses the linear approximation of the tire lateral force of the following form,

$$F_f = C_f \alpha_f, \quad \alpha_f = \delta - \frac{v_y}{v_x} - \frac{a}{v_x} r,$$

$$F_r = C_r \alpha_r, \quad \alpha_r = -\frac{v_y}{v_x} + \frac{b}{v_x} r.$$

The state predictor is based on the discretized version of the linear model.

$$\dot{x} = Ax + Bu + w$$

$$\hat{x}(k+1) = A_d \hat{x}(k) + B_d u(k) + w_d(k)$$

$$y(k) = C \hat{x}(k)$$

where for sampling period T ,

$$A_d = e^{AT}, B_d = \int_0^T e^{A\tau} d\tau B, w_d(k) = \int_0^T e^{A\tau} d\tau w(kT)$$

The model predictive controller is designed to optimize the following cost

$$J(x, u, k) = \sum_{i=1}^N (y(k+i) - \gamma(k+i))^T Q(k+i) (y(k+i) - \gamma(k+i)) + u(k)^T R(k) u(k)$$

where $y(k)$ is the expected relative measurement to the reference path based on the state predictor. $Q(k)$ and $R(k)$ are positive definite matrices to tradeoff between minimizing error between the path and the actuation cost.

Based on the state predictor model, the cost function will be a quadratic function of the controller $u(k)$. The minimum is achieved when $\frac{\partial J}{\partial u(k)} = 0$. The resulting optimal $u(k)$ takes the form of a linear feedback $u(k) = -K(k)x(k) + v(k)$, where

$$K(k) = \frac{\sum_{i=1}^N \left(C \sum_{j=1}^i A_d^{j-1} B_d \right)^T Q(k+i) C A_d^i}{R(k) + \sum_{i=1}^N \left(C \sum_{j=1}^i A_d^{j-1} B_d \right)^T Q(k+i) \left(C \sum_{j=1}^i A_d^{j-1} B_d \right)}$$

$$v(k) = \frac{\sum_{i=1}^N \left(C \sum_{j=1}^i A_d^{j-1} B_d \right)^T Q(k+i) \left(\gamma(k+i) - C \sum_{j=1}^i A_d^{j-1} w_d(k) \right)}{R(k) + \sum_{i=1}^N \left(C \sum_{j=1}^i A_d^{j-1} B_d \right)^T Q(k+i) \left(C \sum_{j=1}^i A_d^{j-1} B_d \right)}$$

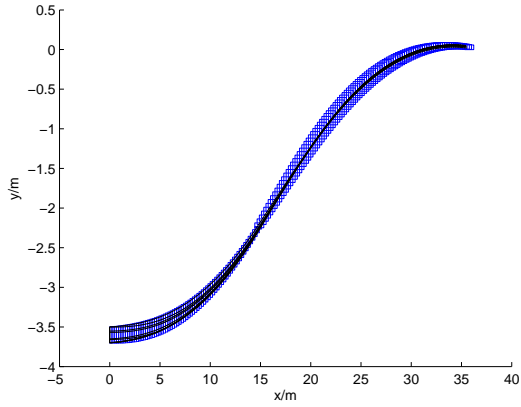
2.4.2 Reachability for Physical System

The *occupancy set* of the vehicle includes the reachable set of the center of mass motion and the physical size of the vehicle. Besides the physical size of the objects, an additional parameter such as safety radius δ_s is considered as well to capture safe proximity of the objects to others. Take a rectangular-shaped object in 2D for example, given the safety radius δ_s , the reachable set R_x, R_y and R_θ for

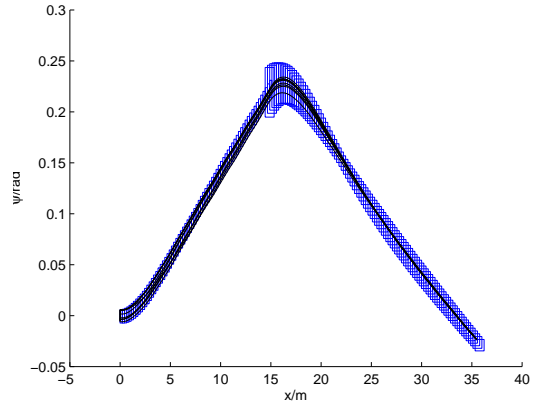
center of mass $[x, y, \theta] \in \mathbb{R}^2 \times [0, 2\pi)$, the occupancy set of the physical object is the set

$$\{(R_x \times R_y) \oplus \mathcal{B}(\delta_s) \oplus \text{Rot}(\theta)([-l/2, l/2] \times [-w/2, w/2]) \quad \theta \in R_\theta\}$$

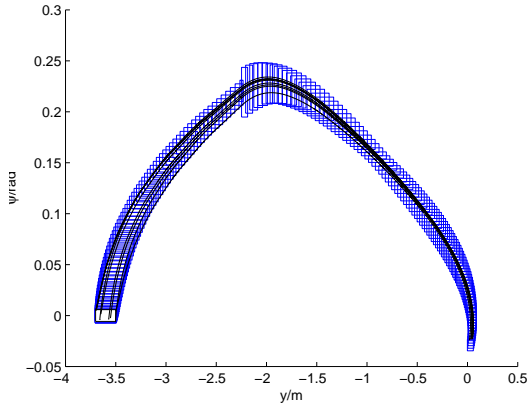
$\text{Rot}(\cdot)$ denote the rotation matrix of its argument, $\mathcal{B}(\cdot)$ denote the closed ball of radius specified by its argument, and w, l are the corresponding width and length of the object.



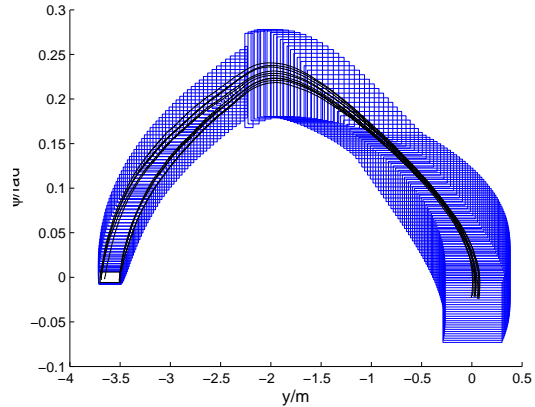
(a) Reachable set projected to longitude distance x and relative lateral distance y .



(b) Reachable set projected to longitude distance x and relative heading ψ .

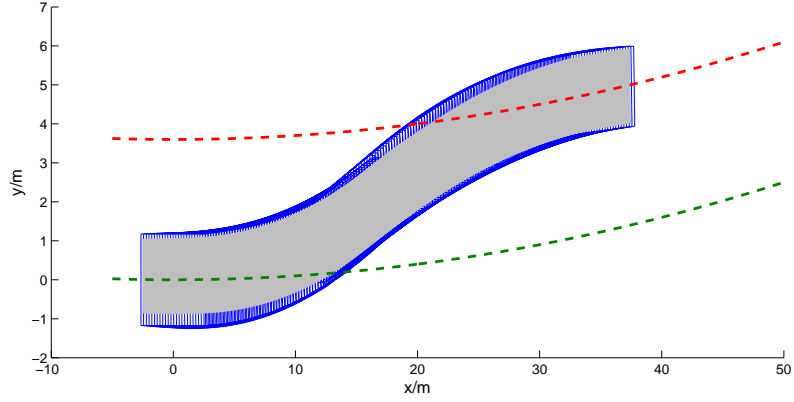


(c) Reachable set projected to relative lateral distance and relative heading.

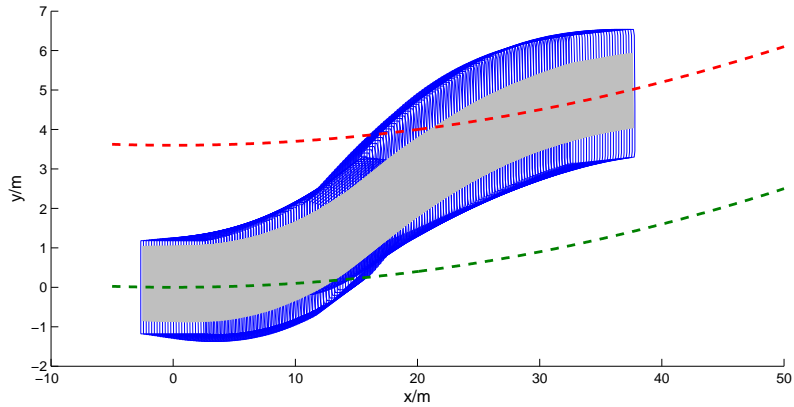


(d) Reachable set after adding deviation on lateral speed and yaw rate.

Figure 2.4: Reachable set of linear hybrid system model. In all the plots, the reachable sets are shown as blue polytopes and simulation traces are black lines. The noise produced a large deviation on the reachable set (Fig. (c) and Fig. (d)). Although the system is still stable, the system is not robust in rejecting noise.



(a) Occupancy set of the vehicle for hybrid linear system model without sensor noise.



(b) Occupancy set of vehicle for hybrid linear system model with sensor noise

Figure 2.5: Clearly the hybrid linear model is not robust in terms of rejecting sensor noise, the occupancy set is enlarged to about 1m around the vehicle. In this particular case, the distance required to avoid collision is reduced for about 2m due to the sensor noise.

2.4.3 Results and Analysis

We performed reachability analysis using existing path planner that generates a collision free path based on distance to the front vehicle and lateral separation requirement. [61] The path planner generates two interconnected arcs as reference signals to the controller. The switching point is when the ego vehicle passes half of the width of the front vehicle. To model this specification, the overall system is

captured in the hybrid system setup. For simplicity, a linear model is used for each segment. The reference signal for the first segment (or the first location of hybrid system) is an arc bending towards the other lane, while for the second segment, the arc guides the car so that it is centered in the target lane and oriented correctly. The initial state uncertainty is specified as following based on system noise. The uncertainty of relative lateral distance is 20cm due to the noise in lane mark detection. The uncertainty of relative heading angle is 1 deg. Without any further disturbance in the system, the reachable set of the vehicle for 1.6s is as shown in Fig. 2.4. In the figures, the reachable sets are covered by blue boxes which depict the reachable set at continuous time intervals. Example simulation traces are the solid black lines. If additional sensor noise is included in the model, more specifically, the state estimates are predicted on noisy state measures such as lateral speed v_y and yaw rate r , the resulting reachable sets are larger comparing to the case without estimation noise, and they do not shrink in size over time (Fig. 2.4d). The specific deviation used in this simulation is 0.5km/h deviation on velocity estimate v_y and 1 deg/s deviation on r . It can be seen based on the reachable sets that the overall system is not robust to sensor noise. The reachable sets do not converge to the ones obtained earlier in Fig. 2.4. The associated occupancy sets of the vehicle with and without sensor noise are captures in Fig. 2.5. The gray area represents the occupancy sets of one simulation run, while the blue boxes cover the occupancy sets based on reachability analysis. The distance required to avoid collision is increased by around 2m due to the sensor noise. The performance of the algorithm scales well with the system complexity. The computation for 1.6 sec reachable set takes about

2 sec in Matlab.

2.5 Conclusion

In this chapter, we examined the safety and robustness of trajectory planner and tracking controller using forward reachability analysis. We showed that the computation is fast, efficient for linear or hybrid system with linear dynamics. We demonstrated usage of the reachability based verification for collision avoidance maneuvers for UAVs and semiautonomous vehicles.

Chapter 3: Control Synthesis for Collision Avoidance Using Reachable Set

3.1 Overview

Autonomous aircraft have been deployed for agriculture research and management, surveillance and sensor coverage for threat detection and disaster search and rescue operations. In most of these scenarios, it is desirable to have multiple aircraft to increase the efficiency and coverage of the UAVs. Since the UAVs in these scenarios, and increasingly in more commercial applications, will be deployed in the shared commercial airspace, they are required to have sophisticated collision avoidance algorithms in order to fly together with other conventional aircraft. As the number of these UAVs increases, a centralized ground control based model is not sufficient alone. Thus an autonomous on-board collision avoidance system needs to be implemented in a decentralized manner and in real-time. As discussed in the previous chapter, reachable sets can be used to compute occupation sets for aircraft in collision course. Therefore, our objective is to synthesize control sets for the aircraft in collisions so that the reachable tubes do not collide at any time. The collision avoidance problem between reachable sets has been previously studied under

the frame of reachability analysis of nonlinear dynamical games [14, 15]. The other agent is considered as adversary or disturbance to the collision avoidance problem. A controller is synthesized to allow the aircraft to avoid the reachable sets of others. However, in most collision avoidance scenarios, the controller selection is collaborative. In this chapter, we investigate cases when agents can derive collaboratively the control constraint sets so that the resulting reachable sets are collision free. The control constraint can be constant or time varying over the horizon, in general the sets form a control tube. The control tube describes the robustness of the control to disturbances and behaviors of the other aircraft. Part of the work is published in [62].

Due to the nature of collision avoidance, it is critical to have fast and efficient computation for reachable sets. There are several fast linear algorithms based on convex analysis for reachability analysis. These algorithms employ linearized dynamics with convex initial state set and control disturbance set. They commonly approximate reachable set using specific covering sets including ellipsoids [16, 17] and polytopes [19–22]. In all these cases, commonly support functions are used to analytically derive the reachable sets. However, many algorithms [19–22] compute the approximated convex set iteratively, which makes the solutions impossible to be represented in analytic forms. We will use the reachable set tool set from [63] based on the ellipsoid methods in [16], because its solutions can be expressed efficiently in analytical expressions.

The main contribution of this part of the work is that we propose and solve a new reachability based formulation of collision avoidance. It is formulated as

the following two-fold optimization problem. In the first part, autonomous aircraft collaboratively define control constraint sets, while in the second, individual aircraft will compute an optimal control policy within the control constraints so that they can reach their objective and avoid the others. We focus on the first part, since the second part is a traditional optimal control problem with hard control constraints. We provide two distinct control tube update policies. The first one is a constant control set design [62]. We focus on collaboration between UAVs and controls such that their reachable tubes are separated under new control set constraint. The second method is a time varying control tube design. We focus on cases when one aircraft is considered as an intruder and its expected states are captured using an estimated reachable tube based on sensors. We then derive time varying robust control tube such that the ego vehicle can avoid the intruder. The rest of this chapter is organized as follows. In section 3.2 we present the fundamentals of reachability sets. Then in section 3.3 we define the reachable set collision avoidance problem and formulate it into a convex optimization problem incorporating the reachable sets. Both time varying and constant control tube design are discussed in detail there. Afterwards, we demonstrate our approach in scenarios involving collision avoidance between quadrotors and fixed-wing aircraft.

3.2 Reachable Set

We consider collision avoidance navigation between aircraft whose dynamics are given by nonlinear models as (3.1).

$$\dot{x}(t) = f(t, x, u, v) \quad (3.1)$$

where $x(t) \in \mathcal{X}$, $x(0) \in \mathcal{X}_0 \subseteq \mathcal{X}$, $u(t) \in \mathcal{U}(t)$ for all t , $v(t) \in \mathcal{V}$ for all t .

Reachable set computation for nonlinear model directly exists, but it is commonly impractical in collision avoidance due to slow computation time [14]. So instead of looking at the full nonlinear model, we first exam the linearized dynamics of nonlinear system around an fixed operating point, resulting in dynamics that are of the following form.

$$\dot{x}(t) = A(t)x(t) + B(t)u(t) + v(t) \quad (3.2)$$

To simplify the computation, the following assumptions are used by noting Lemma 3.2.1, where $\mathcal{U}(t)$, \mathcal{X}_0 and \mathcal{V} are all convex and compact sets. [16]

Lemma 3.2.1. *With $\mathcal{U}(t)$, \mathcal{X}_0 and \mathcal{V} being convex and compact, the reachable set $\mathcal{R}[\vartheta]$ is also convex and compact.*

The problem of defining the reachable set of the system can be reformulated as an optimization problem. Consider the system (3.2). Since the reachable set would be convex and compact due to the assumption on the control set and the initial set, the reachable set can be captured using its support function. Let $\rho(l|X)$ be the support function of the set X , i.e. $\rho(l|X) = \max\{\langle l, x \rangle | x \in X\}$, and $\langle l, x \rangle$

represents the inner product between vector l and x . Then the support function of the reachable set $\mathcal{R}[\vartheta]$ is given by the following,

$$\begin{aligned}
\rho(l|\mathcal{R}[\vartheta]) &= \max\{\langle l, x \rangle \mid x \in \mathcal{R}[\vartheta]\} \\
&= \max \left\{ \int_{t_0}^{\vartheta} l' \Phi(\vartheta, s) B(s) u(s) + l' \Phi(\vartheta, s) v(s) ds \right. \\
&\quad \left. + l' \Phi(\vartheta, t_0) x_0 \mid u(s) \in \mathcal{U}(s), x_0 \in X_0, v(s) \in \mathcal{V} \right\} \\
&= \int_{t_0}^{\vartheta} \rho(B'(s) \Phi'(\vartheta, s) l | \mathcal{U}(s)) ds \\
&\quad + \int_{t_0}^{\vartheta} \rho(\Phi'(\vartheta, s) l | \mathcal{V}) ds + \rho(\Phi'(\vartheta, t_0) l | X_0)
\end{aligned} \tag{3.3}$$

where $\Phi(t, s)$ is the transition matrix of the system (3.2). i.e. it satisfies $\frac{\partial}{\partial t} \Phi(t, s) = A(t) \Phi(t, s)$ and $\Phi(s, s) = \mathbf{I}$. Assume further that all the sets are represented by ellipsoids. Let c_X and M_X denote the center and shape matrix of the set. The following holds, if $x \in X = E(c_X, M_X)$,

$$\langle x - c_X, M_X^{-1}(x - c_X) \rangle \leq 1.$$

In terms of the support function, it can be expressed by

$$\langle l, x \rangle \leq \langle l, c_X \rangle + \langle l, M_X l \rangle^{1/2}.$$

The support function of the reachable set $\mathcal{R}[\vartheta]$ could be expressed further in terms of the centers and shape matrices of the initial set, control and disturbance sets

(equation (3.4)).

$$\begin{aligned}
\rho(l|\mathcal{R}[\vartheta]) &= \langle l, \Phi(\vartheta, 0)c_{X_0} \rangle + \left\langle l, \int_0^\vartheta \Phi(\vartheta, s)B(s)c_{\mathcal{U}}(s)ds \right\rangle \\
&\quad + \langle l, \Phi(\vartheta, 0)M_{X_0}\Phi^T(\vartheta, 0)l \rangle^{1/2} \\
&\quad + \left\langle l, \int_0^\vartheta \Phi(\vartheta, s)c_{\mathcal{V}}(s)ds \right\rangle \\
&\quad + \int_0^\vartheta \langle l, \Phi(\vartheta, s)B(s)M_{\mathcal{U}}B^T(s)\Phi^T(\vartheta, s)l \rangle^{1/2} ds \\
&\quad + \int_0^\vartheta \langle l, \Phi(\vartheta, s)M_{\mathcal{V}}\Phi^T(\vartheta, s)l \rangle^{1/2} ds
\end{aligned} \tag{3.4}$$

Since, as will be discussed later, the disturbance is a constant term to the optimization problem, we will assume in what follows that $v(t) = 0$. The term related to $v(t)$ affects the size of the reachable set. Since it is independent of the control set parameter, it can be treated as an additional separation required in the collision avoidance problem.

The computation of the reachable set for the nonlinear dynamics linearized around a trajectory is similar. Assume the nonlinear system is linearized around a steady operating trajectory $x^*(t)$, with steady state controller $u^*(t)$. Denote total state $X(t) = x^*(t) + x(t)$. The deviation term $x(t)$ is the linearized dynamics, with state space representation $[A(t), B(t)]$ derived from 3.1,

$$A(t) = \left. \frac{\partial f}{\partial x} \right|_{x=x^*}, \quad B(t) = \left. \frac{\partial f}{\partial u} \right|_{u=u^*}$$

The reachable set of the original nonlinear system has the same shape but with additional center offset based on the steady state dynamics $x^*(t)$ which evolves according to the following.

$$\dot{x}^*(t) = f(t, x^*, u^*)$$

In the following section, we assume the system is a linear dynamics. In the simulation section, we provide a nonlinear example. The method is very similar but with few constraints adjusted accordingly due to the steady state dynamics.

3.3 Collision Avoidance Between Two Agents Using Reachability Analysis

3.3.1 Constant Control Set Design for Collaborative Collision Avoidance

Let us consider the following two agents reachability based collision avoidance problem.

Problem 3.3.1 (Reachability Based Collision Avoidance). *We seek a control algorithm for the aircraft A and B, such that they can always avoid each other if each of them uses a more constrained control set than their initial ones. More specifically, in the first phase, given the initial state estimation set of the aircraft B X_0^B , and the estimated control set \mathcal{U}^B , we seek a tighter control constraint set $\tilde{\mathcal{U}}^B$ such that aircraft A can find a safe reachable tube that does not intersect with the one of aircraft B counting the separation. At the same time, we need to make sure that under the new control constraint set $\tilde{\mathcal{U}}^B$, aircraft B can still perform the maneuvers the system requires to reach its goal area. At the second phase we seek a safe reachable tube for aircraft A so that the reachable tube will be apart from the reachable tube of aircraft B for at least the required separation. Afterward we seek trajectories*

within the reachable tube, so that they can safely reach their objectives in an optimal manner.

As assumed in the previous section, let the initial set and control set of aircraft B be the ellipsoids $X_0^B = E(c_{X_0}^B, M_{X_0}^B)$, $\mathcal{U}^B = E(c_U^B, M_U^B)$, and denote by $c_X^A(t)$ the nominal trajectory of aircraft A. We further assume the two aircraft will be closest at time τ in the future, so we can reason about the reachable set instead of the tube. For the first phase we want to keep the reachable set of aircraft B away from the nominal trajectory of A as far as we can, and keep the constrained control set of aircraft B relatively large. This can be formulated into the following optimization problem.

Problem 3.3.2 (Optimization Part I).

$$\begin{aligned} & \max_{q(t), Q(t)} \max_l \quad \langle l, c_X^A(\tau) \rangle - \rho(l|\mathcal{R}_x^B(\tau)) + k \log(\det(Q(t))) \\ & \text{subject to} \quad E(q(t), Q(t)^T Q(t)) \subset \mathcal{U}^B \quad \forall t \in [0, \tau] \\ & \quad \quad \quad \|l\| = 1 \\ & \quad \quad \quad \langle l, c_X^A(\tau) \rangle - \rho(l|\mathcal{R}_x^B(\tau)) \geq 0 \end{aligned}$$

where $\rho(l|\mathcal{R}_x^B(\tau))$ is the support function of the reachable set of aircraft B at time τ :

$$\begin{aligned} \rho(l|\mathcal{R}_x^B(t)) &= \langle l, \Phi(t, 0)c_{X_0}^B \rangle + \left\langle l, \int_0^t \Phi(t, s)B(s)c_U^B(s)ds \right\rangle \\ & \quad + \langle l, \Phi(t, 0)M_{X_0}^B \Phi^T(t, 0)l \rangle^{1/2} \\ & \quad + \int_0^t \langle l, \Phi(t, s)B(s)Q^T(s)Q(s)B^T(s)\Phi^T(t, s)l \rangle^{1/2} ds. \end{aligned} \tag{3.5}$$

The parameters of the optimization are $q(t)$ and $Q(t)$, both related to the new control set. More specifically $\tilde{\mathcal{U}}^B = E(q(t), Q^T(t)Q(t))$. In the objective function,

the first part is the distance between the nominal trajectory $c_X^A(\tau)$ and the reachable set, and the second part is the size of the control set. Scalarization is used to determine the Pareto optimal points [64]. The inner maximization determines distance by varying the direction vector l . The first constraint is due to the fact that $\tilde{\mathcal{U}}^B \subset \mathcal{U}^B$, the last constraint is to keep nominal trajectory outside the reachable set of aircraft B.

$E(q(t), Q(t)^T Q(t)) \subset \mathcal{U}^B$ is equivalent to the following constraints on a new parameter $\lambda > 0$,

$$\begin{bmatrix} 1 - \lambda & 0 & (q(t) - c_U^B)^T \\ 0 & \lambda I & Q(t) \\ q(t) - c_U^B & Q(t) & M_U^B \end{bmatrix} \succeq 0.$$

Let A, B be time invariant, we seek a time invariant control constraint set as well, so q and Q are time invariant. Let us assume the optimal l^* can be estimated based on the initial center of the reachable set alone. In other words, we assume the direction that minimizes distance between the reachable set and $c_X^A(\tau)$ is not affected by changes in the control constraint. The intuition behind this assumption is that even if the direction is altered, the outer maximization is achieved at a similar constraint set. In real applications, the autonomous aircraft will be given such direction to avoid either based on the approaching angle autonomously or based on instructions from the other pilots. Then the overall problem becomes the following.

Problem 3.3.3 (Simplified Problem Part I).

$$\begin{aligned}
& \max_{q, Q} \quad \langle l^*, c_X^A(\tau) \rangle - \rho(l^* | \mathcal{R}_x^B(\tau)) + k(\log \det(Q)) \\
& \text{subject to} \quad \lambda > 0 \\
& \quad \quad \quad \begin{bmatrix} 1 - \lambda & 0 & (q - c_U^B)^T \\ 0 & \lambda I & Q \\ q - c_U^B & Q & M_U^B \end{bmatrix} \succeq 0 \\
& \quad \quad \quad \langle l^*, c_X^A(\tau) \rangle - \rho(l^* | \mathcal{R}_x^B(\tau)) \geq 0
\end{aligned}$$

$$\begin{aligned}
\rho(l | \mathcal{R}_x^B(t)) &= \langle l, e^{At} c_{X_0}^B \rangle + \left\langle l, \int_0^t e^{A(t-s)} ds B q \right\rangle \\
&+ \langle l, e^{At} M_{X_0}^B e^{At} l \rangle^{1/2} \\
&+ \int_0^t \langle l, e^{A(t-s)} B Q^T Q B^T (e^{A(t-s)})^T l \rangle^{1/2} ds.
\end{aligned} \tag{3.6}$$

To make the overall problem a simple convex optimization problem, we note the following two cases.

(i) Suppose Q is $r * (M_U^B)^{1/2}$ i.e. the constrained control set is a scaled version of the initial control set. Then we have the following.

Problem 3.3.4 (Scaled Initial Set Method).

$$\begin{aligned}
& \max_{q,r} && \langle l^*, c_X^A(\tau) \rangle - \rho(l^* | \mathcal{R}_x^B(\tau)) \\
& && + k \dim(M_U^B) \log r \\
& \text{subject to} && \lambda > 0 \\
& && \begin{bmatrix} 1 - \lambda & 0 & (q - c_U^B)^T \\ 0 & \lambda I & r(M_U^B)^{1/2} \\ q - c_U^B & r(M_U^B)^{1/2} & M_U^B \end{bmatrix} \succeq 0 \\
& && \langle l^*, c_X^A(\tau) \rangle - \rho(l^* | \mathcal{R}_{xB}(\tau)) \geq 0
\end{aligned}$$

$$\begin{aligned}
\rho(l | \mathcal{R}_x^B(t)) &= \langle l, e^{At} c_{X_0}^B \rangle + \left\langle l, \int_0^t e^{A(t-s)} ds B q \right\rangle \\
&+ \langle l, e^{At} M_{X_0}^B e^{At} l \rangle^{1/2} \\
&+ r \int_0^t \langle l, e^{A(t-s)} B M_U^B B^T (e^{A(t-s)})^T l \rangle^{1/2} ds.
\end{aligned} \tag{3.7}$$

(ii) Assume now that the requirement for $\langle l^*, c_X^A(\tau) \rangle - \rho(l^* | \mathcal{R}_x^B(\tau)) \geq 0$ is removed. The last term of $\rho(l^* | \mathcal{R}_x^B(\tau))$ can be upper bounded using the matrix norm property. Then the objective function can be lower bounded by $\langle l^*, x_{Ac} \rangle - \tilde{\rho}(l^* | \mathcal{R}_x^B(\tau))$, where

$$\begin{aligned}
\tilde{\rho}(l | \mathcal{R}_x^B(t)) &= \langle l, e^{At} c_{X_0}^B \rangle + \left\langle l, \int_0^t e^{A(t-s)} ds B q \right\rangle \\
&+ \langle l, e^{At} M_{X_0}^B e^{At} l \rangle^{1/2} \\
&+ \|Q\|_2 \int_0^t \langle l, e^{A(t-s)} B B^T (e^{A(t-s)})^T l \rangle^{1/2} ds.
\end{aligned} \tag{3.8}$$

As the result, we have the following convex optimization problem.

Problem 3.3.5 (Matrix Norm Method).

$$\begin{aligned}
& \max_{q, Q} && \langle l^*, c_X^A(\tau) \rangle - \tilde{\rho}(l^* | \mathcal{R}_x^B(\tau)) \\
& && + k \log(\det(Q)) \\
& \text{subject to} && \lambda > 0 \\
& && \begin{bmatrix} 1 - \lambda & 0 & (q - c_U^B)^T \\ 0 & \lambda I & Q \\ q - c_U^B & Q & M_U^B \end{bmatrix} \succeq 0 \\
& && \langle l^*, c_X^A(\tau) \rangle - \tilde{\rho}(l^* | \mathcal{R}_x^B(\tau)) \geq 0
\end{aligned}$$

Both methods can be solved easily by a convex optimization solver in particular a semidefinite programming solver. We use CVX [65] in the demonstration discussed in a later section. After the control set of aircraft B is determined, aircraft A can define its control set so that the reachable set does not collide with the safe set of B. The safe set of B is the reachable set of B enlarged by the required separation between aircraft A and B. The difference with the previous problem is the fact that we cannot use the center of aircraft A to maximize the distance to the reachable set of B. Instead we add a constraint to keep the distance between the sets larger than zero. If such a problem is feasible, standard optimization algorithms will be used to obtain control laws for both aircraft under the modified constrained control set. If it is not feasible, then it means the scalarization factor is too large. By iteratively decreasing the scalarization factor, one can find a pair of good control sets that keep the reachable set of both aircraft balanced. After they avoid each other on the closest contact point, the original navigation will resume.

The optimization part II is addressing the problem to obtain the largest control

set for aircraft A. After the constrained control set of aircraft B is determined, the external ellipsoid approximation of the reachable set of aircraft B will be computed based on the ellipsoid toolbox [63]. The external approximation is given as an intersection of ellipsoids with the same center. The safe set of B is defined as the Minkowski sum of the reachable set of aircraft B and a ball, whose radius is the required separation. It can be computed by the ellipsoid toolbox as the intersection of overapproximated ellipsoids as well. For simplicity, we take an external ellipsoid approximation of the intersection set as the safe set of B. Let such overapproximation ellipsoid be $E(c_X^B, M_X^B)$. Assume that the direction for which the minimum distance is achieved is l^* as assumed earlier. Then the largest control set for aircraft A that satisfies the safety requirement can be determined by the following optimization problem.

Problem 3.3.6 (Optimization Part II).

$$\begin{aligned}
& \max_{q, Q} && \log(\det(Q)) \\
& \text{subject to} && E(q, Q^T Q) \subset \mathcal{U}^A \\
& && -\rho(-l^* | \mathcal{R}_x^A(\tau)) - \rho(l^* | E(c_X^B, M_X^B)) > 0
\end{aligned}$$

This can be again transformed into a convex optimization problem by using the shrinking initial set or the norm method. We will focus on the norm method for this part. The last constraint can be reformulated into the following:

$$\begin{aligned}
& \langle l^*, e^{A\tau} c_{X_0}^A \rangle + \left\langle l^*, \int_0^\tau e^{A(\tau-s)} ds B q \right\rangle - \langle l^*, c_X^B \rangle \\
& - \langle l^*, e^{A\tau} M_{X_0}^A e^{A\tau} l^* \rangle^{1/2} - \langle l^*, M_X^B l^* \rangle^{1/2} \\
& - \|Q\|_2 \int_0^t \langle l^*, e^{A(\tau-s)} B B^T (e^{A(\tau-s)})^T l^* \rangle^{1/2} ds > 0.
\end{aligned}$$

3.3.2 Time Varying Control Tube Design for Collision Avoidance

In the cases of uncollaborative UAVs, the predicted path of the intruder is commonly known apriori. Furthermore, the uncertainty of the intruder can be predicted through online sensors or ground radar. The main problem we want to address, is to generate an update policy for the control set, so that the reachable tube of the ego aircraft is collision free from the predicted tube of the intruder.

The control set under the ellipsoid formulation is described by

$$\mathcal{U}(t) = E(c_{\mathcal{U}}(t), M_{\mathcal{U}}(t)).$$

From previous section, we find out that by scaling the control set we obtain a convex problem (Prob. 3.3.4). To simplify the parameterization and formulation of the optimization problem, we define the control set by a scaling factor $\alpha_{\mathcal{U}}(t)$ of the original system control constraint and the nominal control $c_{\mathcal{U}}(t)$, i.e. $\mathcal{U}(t) = E(c_{\mathcal{U}}(t), \alpha(t)M_{\bar{\mathcal{U}}})$. $\bar{\mathcal{U}} = E(c_{\bar{\mathcal{U}}}, M_{\bar{\mathcal{U}}})$ is an ellipsoid inner-approximation of the physical control limitations. Naturally we have the following constraint on $c_{\mathcal{U}}(t)$ and $\alpha(t)$.

$$E(c_{\mathcal{U}}(t), \alpha(t)M_{\bar{\mathcal{U}}}) \subseteq \bar{\mathcal{U}} \quad \forall t$$

More specifically, this collision avoidance problem will be defined as the following problem involving reachable tubes.

Problem 3.3.7 (Reachability Based Collision Avoidance). *Denote the estimated collision time as T . The collision avoidance using reachability can be formulated as a two steps optimization problem. In the first step, We seek a control set policy $\mathcal{U}(t)$*

over the time interval, $[0, T]$, for the ego aircraft, such that the resulting reachable tube $\mathcal{R}_x^e([0, T])$ does not intersect with the intruder aircraft reachable tube $\mathcal{R}_x^i([0, T])$ counting the separation. These control sets should be chosen to maximize the flexibility of later Model Predictive Control (MPC) design, therefore, the objective will be to maximize the size of the control sets over time $[0, T]$. In the second step, we seek controls within these control sets, so that the ego aircraft can safely reach their objectives in an optimal manner.

We will focus mainly on the first step of Problem 3.3.7 in this part of the thesis. As assumed in the previous section, let the initial state and control set of ego aircraft be the ellipsoids $X_0^e = E(c_{X_0^e}, M_{X_0^e})$ and $\bar{U} = E(c_{\bar{U}}, M_{\bar{U}})$ respectively. Formally, the Problem 3.3.7 can be formulated as the following optimization problem,

Problem 3.3.8 (Reachable Set Collision Avoidance).

$$\begin{aligned}
 & \max_{q(t), \alpha(t)} \max_{l(t)} && \int_{t=0}^T \mu(t) \alpha(t) dt \\
 & \text{subject to} && \forall t \in [0, T] : \\
 & && E(q(t), \alpha(t) M_{\bar{U}}) \subseteq E(c_{\bar{U}}, M_{\bar{U}}) \\
 & && -\rho(-l(t) | \mathcal{R}_x^i(t)) - \rho(l(t) | \mathcal{R}_x^e(t)) > 0
 \end{aligned}$$

The parameters of the optimization are $q(t)$ and $\alpha(t)$, both related to the updated control sets, $\tilde{U}(t) = E(q(t), \alpha(t) M_{\bar{U}})$. The objective is to maximize the sizes of the control sets over time. $\mu(t)$ is a fixed scalar function which specifies the importance of control sets over time. If the weight is uniform over time, the optimal solution can have very flexible control set at the start, but very tight control near collision, which is not desired. The inner maximization is a feasibility problem,

which is to find the direction for a series of separation hyperplanes $l(t)$ induced by the reachable set separation constraint. The first constraint is due to the fact that $\tilde{\mathcal{U}}(t) \subset \bar{\mathcal{U}}$, the last constraint is to keep reachable sets separated at every time step.

The first constraint is equivalent to the following constraints [63] on a new function $\lambda(t) > 0, \forall t \in [0, T]$, such that

$$\begin{bmatrix} 1 - \lambda(t) & 0 & (q(t) - c_{\bar{\mathcal{U}}})^T \\ 0 & \lambda(t)I & a(t)(M_{\bar{\mathcal{U}}})^{1/2} \\ q(t) - c_{\bar{\mathcal{U}}} & a(t)(M_{\bar{\mathcal{U}}})^{1/2} & M_{\bar{\mathcal{U}}} \end{bmatrix} \succeq 0$$

$$a(t)^2 = \alpha(t)$$

Let us assume the norm of the best separation hyperplane $l^*(t)$ can be estimated based on the initial predicted tube of the intruder and ego aircraft. In other words, we assume the direction that minimizes distance between the reachable sets is not affected by changes in the control constraint. The intuition behind this assumption is that even if the direction is altered, the outer maximization is achieved at a similar constraint set. In real applications, the autonomous aircraft will be given such direction to avoid either based on the approaching angle autonomously or based on instructions from the other pilots. Let us define ellipsoids $E(c_X^i(t), M_X^i(t)) \supseteq \mathcal{R}_x^i(t), t \in [0, T]$, which tightly over-approximate the reachable tube of the intruder.

The last constraints can be written as

$$\begin{aligned}
& \langle l^*(t), c_X^i(t) \rangle - \langle l^*(t), M_X^i(t) l^* \rangle^{1/2} - \langle l^*(t), e^{At} c_{X_0}^e \rangle \\
& - \underbrace{\left\langle l^*(t), \int_0^t e^{A(t-s)} B q(s) ds \right\rangle}_{h(q,t)} - \langle l^*(t), e^{At} M_{X_0}^e e^{At} l^*(t) \rangle^{1/2} \\
& - \underbrace{\int_0^t \overbrace{\langle l^*(t), e^{A(t-s)} B M_{\bar{U}} B^T (e^{A(t-s)})^T l^*(t) \rangle^{1/2}}^{r(s,t)} a(s) ds}_{g(a,t)} > 0. \quad (3.9)
\end{aligned}$$

Furthermore, we assume the control set is only allowed to be updated every δt and collision happened at N steps in the future, i.e. $N\delta t = T$. We have

$$\mathcal{U}(s) = q_d(k) \oplus \alpha_d(k) \bar{\mathcal{U}} \quad \forall s \in [k\delta t, (k+1)\delta t)$$

Therefore the optimization variables can be captured in terms of $\mathbf{a} \in \mathbb{R}^N$, $\mathbf{q} \in \mathbb{R}^{m \times N}$ such that

$$\mathbf{a}_i = (\alpha_d(i-1))^{1/2} = a_d(i-1)$$

and the i th column of \mathbf{q} is

$$\mathbf{q}_i = q_d(i-1).$$

.

It is also desired to keep the control set varying smoothly on time. Therefore, we also add an term in objective to minimize variation in the nominal controller

$q_d(k)$. Define constant matrix H as

$$H = \begin{bmatrix} 1 & -1 & 0 & 0 & \dots & 0 \\ 0 & 1 & -1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 1 & -1 & 0 \\ 0 & \dots & 0 & 0 & 1 & -1 \end{bmatrix}$$

The variation in nominal control can be captured using $\|H^T \mathbf{q}\|_F$

Let $\{P(i), i = 1, 2, \dots, N\}$ to be N $n \times m$ constant matrices, such that the j th column of $P(i)$ is defined as

$$P(i)_j = \int_{(i-1)\delta t}^{i\delta t} e^{A(k\delta t-s)} b_j ds,$$

where b_j is the j th column of B matrix.

Then $h(q, t)$ and $g(\alpha, t)$ in the separation constraints Eqn. (3.9) evaluated at $k\delta t$ become

$$\begin{aligned} h(\mathbf{q}, k\delta t) &= \left\langle l^*(k\delta t), \sum_{i=1}^k \int_{(i-1)\delta t}^{i\delta t} e^{A(k\delta t-s)} B \mathbf{q}_i ds \right\rangle \\ &= \left\langle l^*(k\delta t), \sum_{i=1}^k \sum_{j=1}^m \int_{(i-1)\delta t}^{i\delta t} e^{A(k\delta t-s)} b_j ds \mathbf{q}_{ji} \right\rangle \\ &= \sum_{i=1}^k l^{*T}(k\delta t) P(i) \mathbf{q}_i \end{aligned} \quad (3.10)$$

$$g(\mathbf{a}, k\delta t) = \sum_{i=1}^k \mathbf{a}_i \int_{(i-1)\delta t}^{i\delta t} r(s, k\delta t) ds \quad (3.11)$$

Let $\boldsymbol{\mu}_i = \mu(i\delta t)$.

Since \mathbf{a} is positive the objective of maximizing $\mu(t)^T \alpha(t)$, is modified to maximize $\boldsymbol{\mu}^T \mathbf{a}$ instead, then the optimization problem 3.3.8 becomes the following,

Problem 3.3.9 (Simplified Problem).

$$\begin{aligned}
& \max_{\mathbf{a}, \mathbf{q}} && \boldsymbol{\mu}^T \mathbf{a} - \|H^T \mathbf{q}\|_F \\
& \text{subject to} && \forall k \in 1, 2, \dots, N : \\
& && \lambda(k) > 0 \\
& && \begin{bmatrix} 1 - \lambda(k) & 0 & (\mathbf{q}_k - c_{\bar{U}})^T \\ 0 & \lambda(k)I & \mathbf{a}_k(M_{\bar{U}})^{1/2} \\ \mathbf{q}_k - c_{\bar{U}} & \mathbf{a}_k(M_{\bar{U}})^{1/2} & M_{\bar{U}} \end{bmatrix} \succeq 0 \\
& && \rho(-l(k\delta t)|\mathcal{R}_x^i(k\delta t)) - \rho(l(k\delta t)|\mathcal{R}_x^e(k\delta t)) > 0
\end{aligned}$$

The last constraint will be affine on \mathbf{q}, \mathbf{a} based on Equation (3.10) and (3.11).

Therefore this problem can be solved using convex optimization.

3.4 Simulations and Results

The reachable set based methods described above are demonstrated on the linearized quadrotor models and fixed-wing models described below. The original nonlinear model of the quadrotor dynamics can be found in Section 2.3.1

3.4.1 Quadrotor Model

To capture the dynamics of the quadrotor properly, we need two coordinate frames. One of them is a fixed frame and will be named as the earth frame, and the second one is the body frame which moves with the quadrotor. The transformation matrix from the body frame to the earth frame is $R(t)$. The quadrotor dynamics has twelve state variables $(x, y, z, v_x, v_y, v_z, \phi, \theta, \psi, p, q, r)$, where $\xi = [x, y, z]^T$ and

$v = [v_x, v_y, v_z]^T$ represent the position and velocity of the quadrotor w.r.t the body frame. (ϕ, θ, ψ) are the roll, pitch and yaw angles, and $\Omega = [p, q, r]^T$ are the rates of change of roll, pitch and yaw respectively.

The Newton-Euler formalism for the quadrotor rigid body dynamics in earth fixed frame is given by:

$$\begin{aligned}
 \dot{\xi} &= v \\
 \dot{v} &= -g\mathbf{e}_3 + \frac{F}{m}R\mathbf{e}_3 \\
 \dot{R} &= R\hat{\Omega} \\
 \dot{\Omega} &= J^{-1}(-\Omega \times J\Omega + u)
 \end{aligned} \tag{3.12}$$

where g is the acceleration due to gravity, $\mathbf{e}_3 = [0, 0, 1]^T$, F is the total lift force and $u = [u_1, u_2, u_3]^T$ are the torques applied. F and u are the control inputs. J is the moment of inertial details on the quadrotor dynamics can be found in [66], [67]. For this work, we linearize the dynamics (3.12) about the hover with yaw constraint to be zero, as it has been done in [68]. Since ψ is constrained to be zero, we remove ψ and r from our system and make the system ten dimensional. Consequently, we only need three control inputs, F, u_1 , and u_2 for the system. The linearized model is the same as what is done in [68], [7]. The system matrices for the linearized model are:

$$A = \begin{bmatrix} \mathbf{0} & I & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \begin{bmatrix} 0 & g \\ -g & 0 \\ 0 & 0 \end{bmatrix} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & I \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}; B = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \begin{bmatrix} 0 \\ 0 \\ 1/m \end{bmatrix} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & I_{2 \times 3} J^{-1} \end{bmatrix}; I_{2,3} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

All zero and identity matrices in A and B are of proper dimensions.

3.4.2 Fix-wing Dynamics Model

Similar to the dynamics of the quadrotor, we used the earth frame and the body frame. Let (ϕ, θ, ψ) be roll pitch yaw similar as before and (p, q, r) be the body frame rotation rates. The transformation matrix from the body frame to the earth frame is R . The velocity state is commonly represented in the body frame as $v_b = (u, v, w)$.

$$\begin{aligned} \dot{\xi} &= Rv_b \\ \dot{v}_b &= \begin{pmatrix} rv - qw \\ qw - ru \\ qu - pv \end{pmatrix} + \frac{1}{m} \begin{pmatrix} F_x \\ F_y \\ F_z \end{pmatrix} \end{aligned} \quad (3.13)$$

$$\dot{R} = R\hat{\Omega}$$

$$\dot{\Omega} = J^{-1}(-\Omega \times J\Omega + T)$$

where (F_x, F_y, F_z) is the force in body frame. T is the torque in body frame. J is the moment of inertial. The force and torque is further related to airspeed and actu-

ator. Due to the decoupling between lateral and longitude motion in conventionally aircraft, we will focus on longitude motion here. The control signal for longitude motion are elevators control δ_e and propeller thrust δ_t . We can obtain the following linearized dynamics around the leveled cruise mode x^* and u^* . In the leveled cruise mode, there is no turning or climbing. The longitude motion can be captured in a 6 state system (x, z, u, w, q, θ) and 2 control (δ_e, δ_t) . The linearized dynamics can be represented in the matrix form.

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \sin \theta^* & -\cos \theta^* & 0 & u^* \cos \theta^* + w^* \sin \theta^* \\ 0 & 0 & X_u & X_w & X_q & -g \cos \theta^* \\ 0 & 0 & Z_u & Z_w & Z_q & -g \sin \theta^* \\ 0 & 0 & M_u & M_w & M_q & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ X_{\delta_e} & X_{\delta_t} \\ Z_{\delta_e} & 0 \\ M_{\delta_e} & 0 \\ 0 & 0 \end{bmatrix}$$

All the terms of the linear dynamics are derived from aerodynamic models that can be found in [69].

3.4.3 Reachable Sets and Constrained Control Sets for Constant Control Set Method

This section summarizes results generated from constant control set method described in Section 3.3.1.

We computed the reachable sets and performed convex optimization using a computer with a 3.4GHz processor and 8GB memory. The software we used include the Ellipsoid toolbox [63] and CVX [65, 70] for solving convex optimization problems.

3.4.3.1 Quadrotor Example

The first scenario is the following: two quadrotors are approaching each other from coordinates around $(1.6, 0.5, 0)\text{m}$ and $(0, 0, 0)\text{m}$ with initial speed $(-0.2, 0, 0)\text{m/s}$, and $(0.2, 0, 0)\text{m/s}$ respectively. The separation requirement is 1m. It is fairly easy to estimate the collision time which is $\tau = 4\text{s}$, and the closest direction l^* . The initial reachable tube of both aircraft projected to the x, y and time axis is shown in Fig. 3.2. The reachable sets at time τ are shown in Fig. 3.1. As can be seen, the reachable sets clearly overlap with each other. There are no guarantees that one can obtain from this initial setup. By varying k in the optimization problem 3.3.2, we get the following pairs of control sets for aircraft A and B. (Fig. 3.3, 3.5). The control sets are obtained using the matrix norm method. The corresponding resulting reachable tubes for agent A and B are plotted in Fig. 3.4, 3.6. Clearly, the resulting reachable tubes avoid each other, and a closer examination shows that

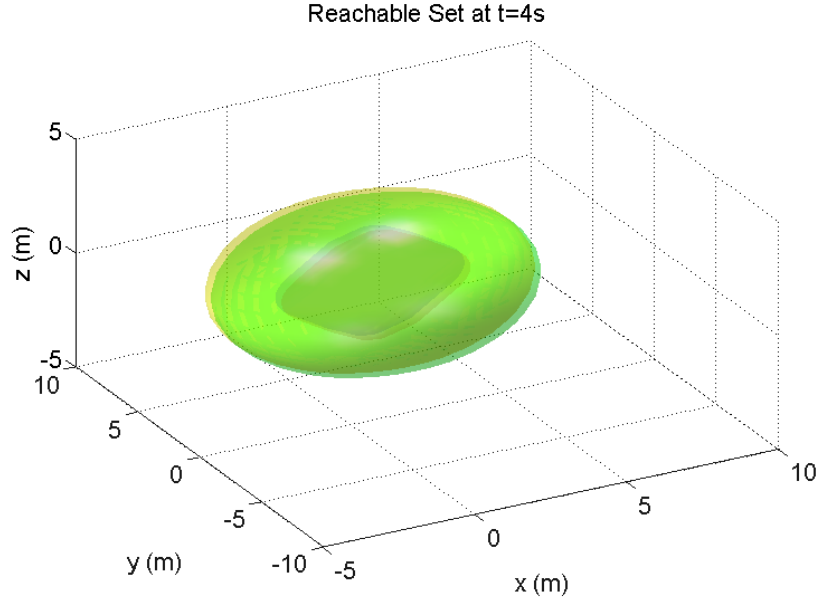


Figure 3.1: The initial reachable sets of both aircraft projected to $x y z$ axis at time τ . The two reachable sets are represented by the internal and external approximations. The reachable sets largely overlap each other. The light colored sets are the external approximations of the reachable sets of aircraft A and B. The internal approximations are the darker color ones inside.

the separation requirement is satisfied.

The computation time for the overall problem is dominated by the reachable set computation. Since we would like to have good precision of the reachable set, 30 ellipsoids are used to obtain the external approximation for both aircraft. The computation for all the reachable sets takes around 430s. In this case study, the collision time is almost imminent (4s). It is only possible to implement this in real time by using a poor estimate of the reachable set with lower precision in this setup. The constrained control sets obtained that way are much smaller. It is also possible to compute the reachable sets in parallel so the resulting computation time would be almost 4s. There are other ways to present the approximation such as via zonotope [19] or support function [20]. Preliminary testing using a zonotope based method

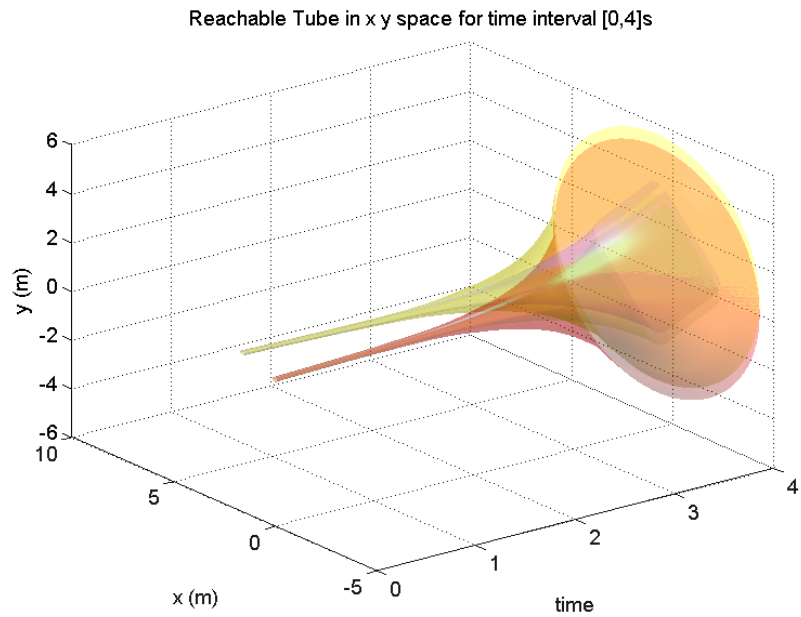


Figure 3.2: The initial reachable tubes of of both aircraft projected to $x y$. Again the reachable tubes are represented by the internal and external approximations. The light yellow tube is the external approximation of the reachable tube of aircraft A, while the light red one is the external approximation of the reachable tube of aircraft B. The internal approximations are the darker color ones inside. Clearly, the reachable tubes collide.

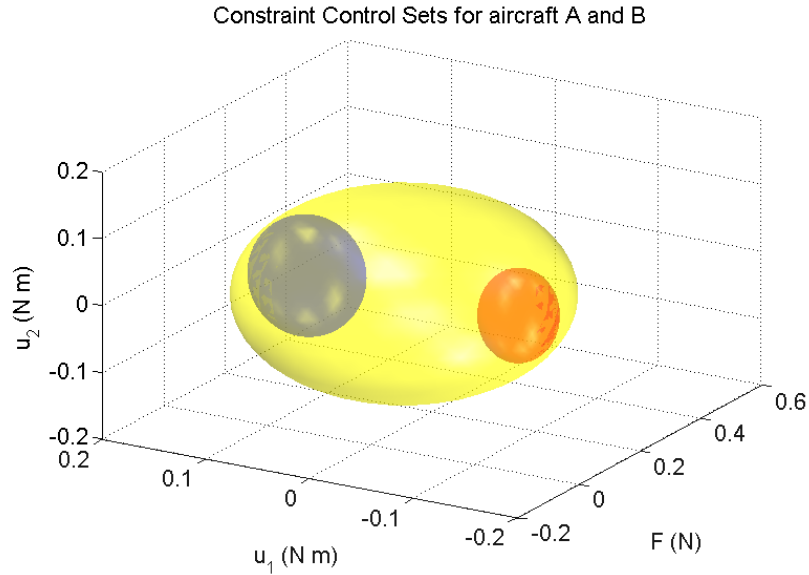


Figure 3.3: This shows the initial control set (yellow ellipsoid), the constrained control set for aircraft A (red ellipsoid), and the constrained control set for aircraft B (blue ellipsoid). This pair of constrained control sets is obtained when the scalarization factor is 1. The control set for u_1 , which contributes to yaw rotation, of aircraft B is larger, i.e. this fits the case when aircraft B has higher priority, so it has more freedom in terms of maneuvers.

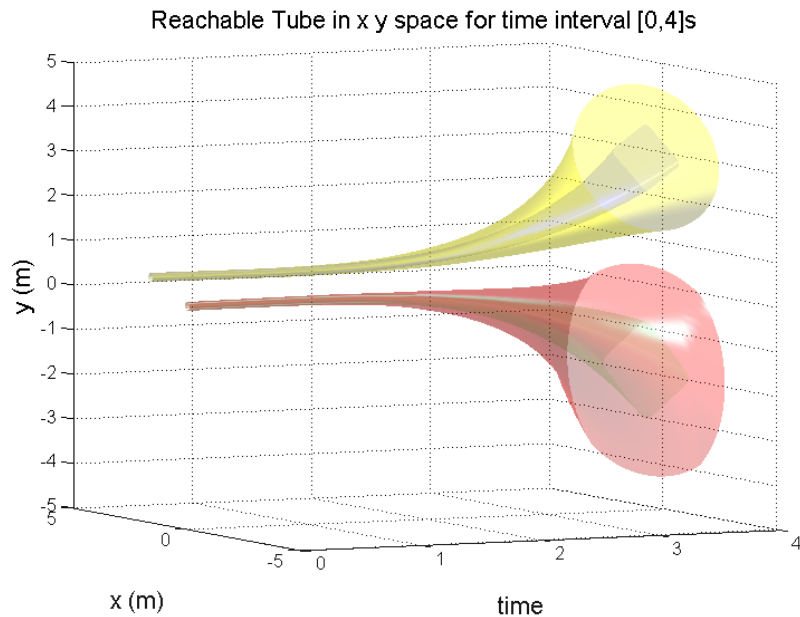


Figure 3.4: The reachable tubes for aircraft A and B with $k = 1$. Clearly, there is no collision between the overapproximation of the reachable tubes. As can be seen from the reachable tube as well, the aircraft A has less freedom comparing to aircraft B. A closer examination also reveals that there is no violation of separation requirement over time.

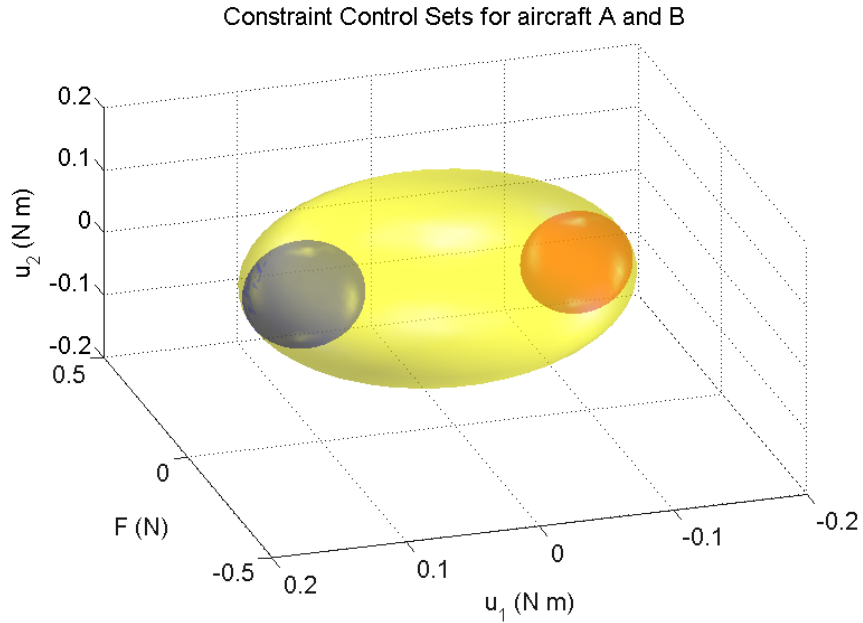


Figure 3.5: This shows the initial control set (yellow ellipsoid), the constrained control set for aircraft A (red ellipsoid), and the constrained control set for aircraft B (blue ellipsoid). This pair of constrained control sets is obtained when the scalarization factor is 0.9. The control set sizes are rather balanced.

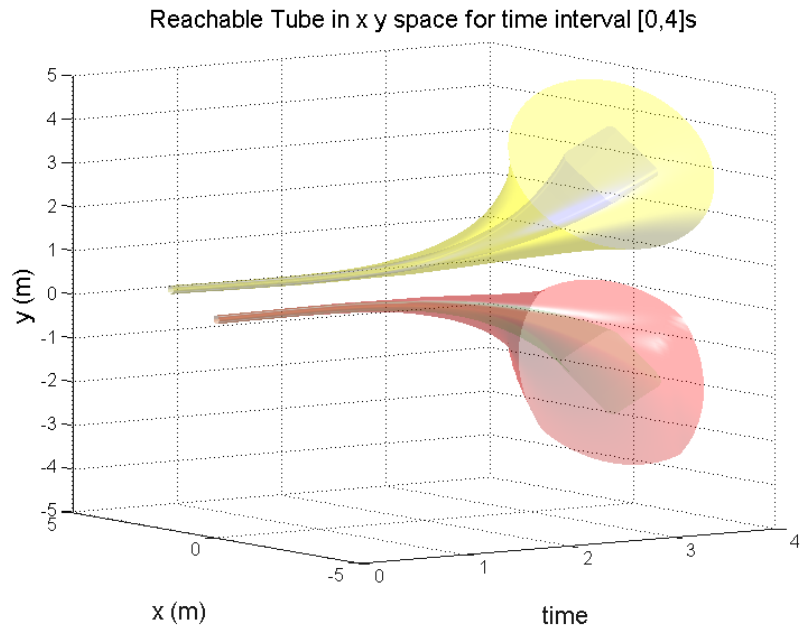


Figure 3.6: The reachable tubes for aircraft A and B with $k = 0.9$. Clearly, there is no collision between the overapproximation of the reachable tubes. A closer examination also reveals that there is no violation of separation requirement over time.

shows faster computation time, but we have not obtained a good representation of the constrained optimization problem 3.3.2 and 3.3.6 using zonotopes. One of our future directions is to use other representations to make the process faster.

However, if one does not require such reachable sets to be computed for real time verification, the computation of the control sets by itself takes very little amount of time. 4s is definitely enough to derive such control sets through convex optimization.

3.4.3.2 Fix-wing Aircraft Example

The second scenario is the following: two fix-wing aircraft are approaching each other from coordinates around $(320, 10, 0)\text{m}$ and $(0, 0, 0)\text{m}$ with initial speed $(-16, 0, 0)\text{m/s}$, and $(16, 0, 0)\text{m/s}$ respectively. The separation requirement is 10m. The collision time is approximately $\tau = 10\text{s}$, and the closest direction l^* is in the z axis. The initial reachable tube of both aircraft projected to the x, z and time axis is shown in Fig. 3.7. As can be seen, the reachable sets clearly overlap with each other. There are no guarantees that one can obtain from this initial setup. By varying k in the optimization problem 3.3.2, we get the following pair of control sets for aircraft A and B. (Fig. 3.8). The control sets are obtained using the matrix norm method. The corresponding resulting reachable tubes for agent A and B are plotted in Fig. 3.9. Clearly, the resulting reachable tubes avoid each other, and a closer examination shows that the separation requirement is satisfied.

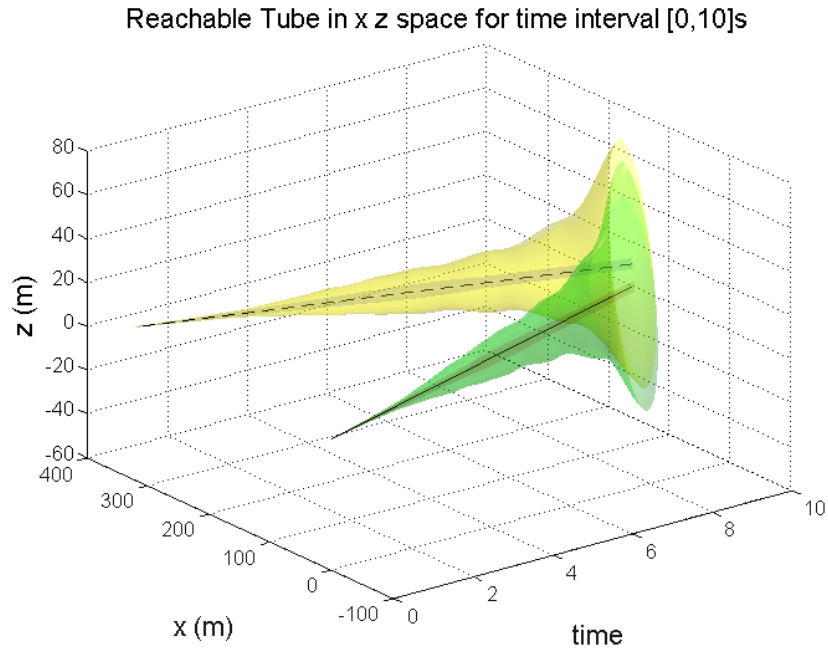


Figure 3.7: The initial reachable tubes of of both aircraft projected to $x z$. The reachable tubes are represented by the internal and external approximations. The light yellow tube is the external approximation of the reachable tube of aircraft A, while the light green one is the external approximation of the reachable tube of aircraft B. The internal approximations are the darker color ones inside. Clearly, the reachable tubes collide. The steady state trajectory for aircraft A is specified by dashed black line, and the steady state trajectory of aircraft B is specified by the blue line.

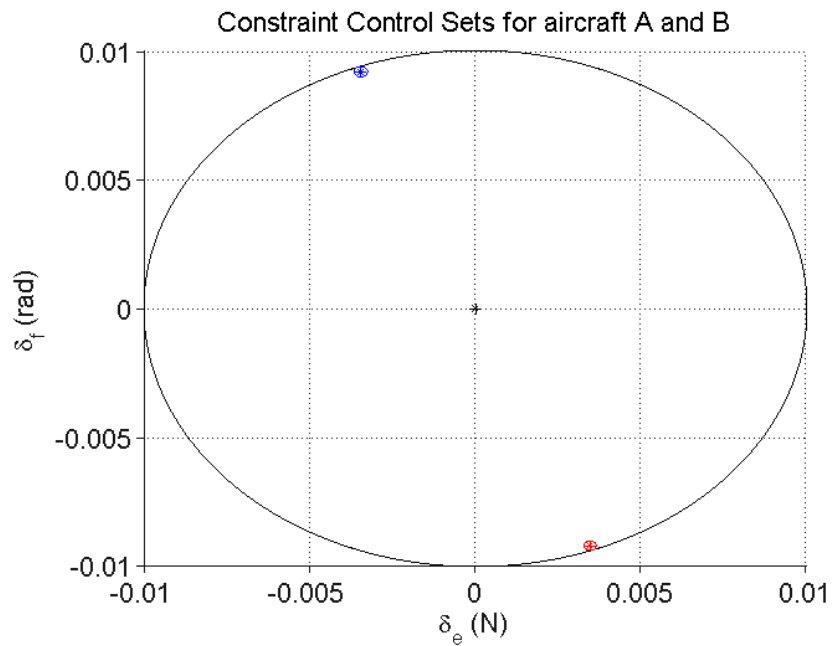


Figure 3.8: This shows the initial control set (black ellipse), the constrained control set for aircraft A (red ellipse), and the constrained control set for aircraft B (blue ellipse). The ellipses are specified by centers and boundaries with their corresponding colors. The constrained control sets are much smaller comparing to the original one, so only center of the ellipse are very clear.

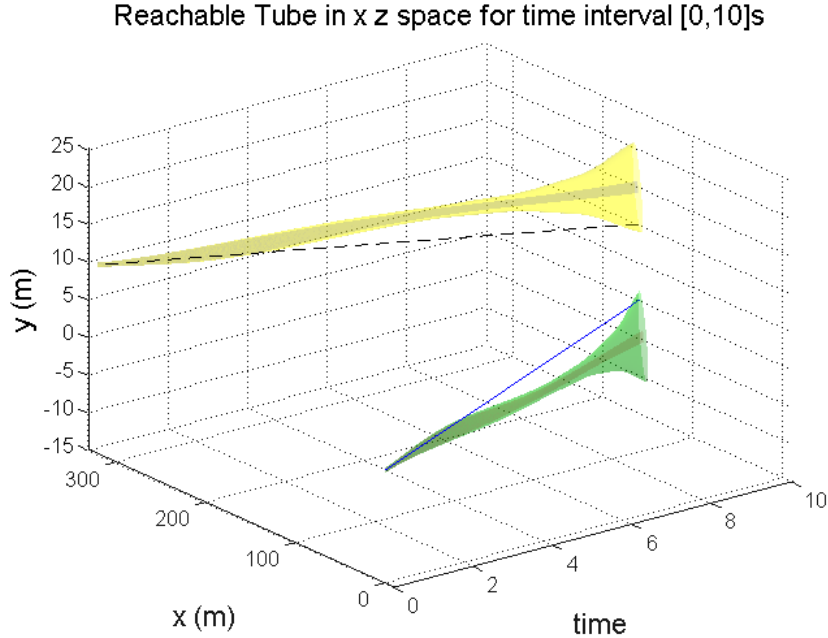


Figure 3.9: The reachable tubes for aircraft A and B. Clearly, there is no collision between the overapproximation of the reachable tubes. A closer examination also reveals that there is no violation of separation requirement over time.

3.4.4 Reachable Sets and Constrained Control Sets for the Time Varying Control Tube Method

This section summarizes result based on the time varying control set method described in Section 3.3.2.

The computation platform and tools are the same as in the previous section. The scenario is the following: the ego aircraft starts at $(-1, 0.5, 0)$ m with the initial speed of $(0.2, 0, 0)$ m/s. The intruder aircraft is currently at $(1, 0, 0)$ approaching the ego aircraft with the initial speed of $(-0.2, 0, 0)$ m/s. The separation requirement is 1m. The collision time $T = 5$ s. The directions of the hyperplane $l^*(t)$ are estimated based on nominal trajectories. The initial reachable tubes of both aircraft projected

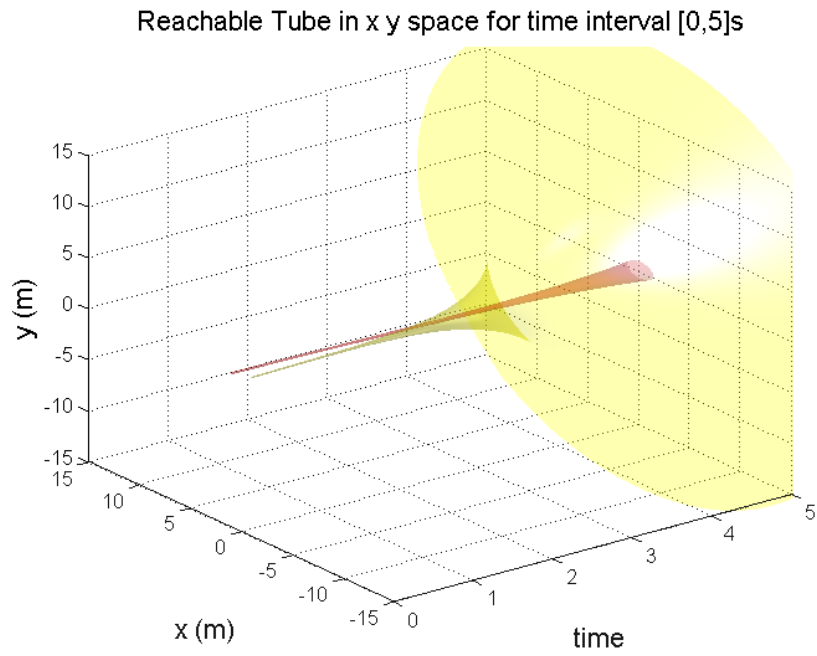


Figure 3.10: The initial reachable tubes of of both aircraft projected to $x y$. The reachable tubes are represented by external approximations computed by Ellipsoid Toolbox. The light yellow tube is the external approximation of the reachable tube of the ego aircraft, while the light red one is the external approximation of the reachable tube of the intruder. Clearly, the reachable tubes collide.

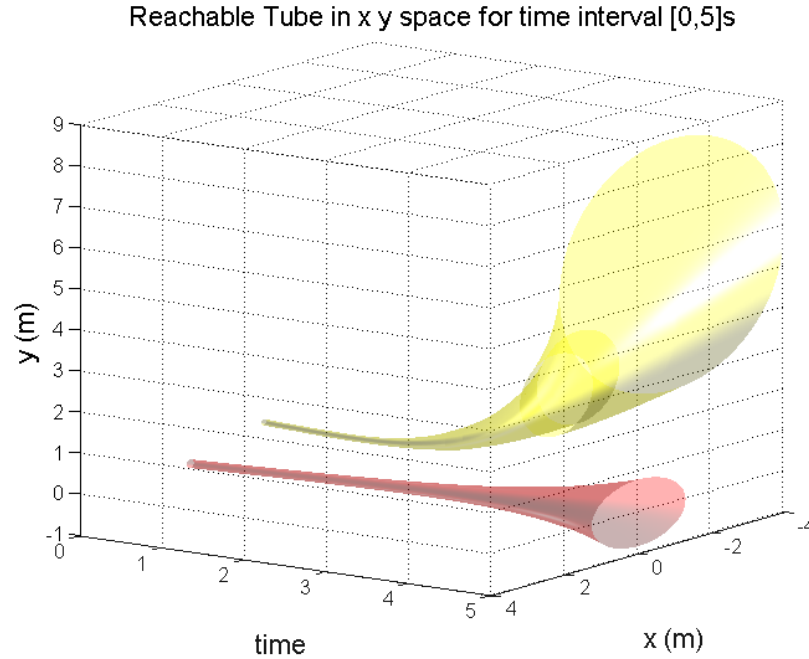


Figure 3.11: The reachable tubes for two aircraft after control set design. Clearly, there is no collision between the overapproximation of the reachable tubes. A closer examination also reveals that there is no violation of separation requirement over time.

to the x , y and time axis are shown in Fig. 3.10. As can be seen, the reachable tubes clearly overlap with each other. By solving the optimization problem, we obtained the resulting reachable tubes for intruder and ego aircraft shown in Fig. 3.11. The newly computed control set is updated every 0.25s. Clearly, the resulting reachable tubes avoid each other, and a closer examination shows that the separation requirement is satisfied.

The computation time for the control set design through convex optimization is about 1.2s, which means the collision avoidance algorithm can be implemented online real-time. The reachable set computation takes a lot more time. Since we would like to have good precision of the reachable set, 10 ellipsoids are used to obtain the external approximation. Given time varying control sets the reachable

tube computation takes around 620s.

3.5 Conclusion

In this chapter, we have proposed several reachability based approaches to collision avoidance of UAVs so that the resulting reachable tubes are collision free. Our approaches provide new insight to the collision avoidance problem in particular regarding collision avoidance of set of trajectories instead of one. Furthermore, as this approach gives limited constraints on the controller, standard optimization based or rule based controller design can be used after our method to obtain optimal and safe trajectories. Our approach reformulates the collision avoidance problem to a two-fold convex optimization problem, which can be solved very efficiently.

We focused the analysis for two aircraft collision avoidance, but our method can be extended to the multiple aircraft case fairly easily by iteratively fixing the control set and the resulting reachable set one by one.

Chapter 4: Optimization based Temporal Logic Planning

4.1 Overview

Autonomous aircraft have been deployed for agriculture research and management, surveillance and sensor coverage for threat detection and disaster search and rescue operations. All these applications require the tasks to be performed in an optimal manner with specific timing constraints. The high level task specifications for these applications generally consist of temporal ordering of subtasks, motion sequencing and synchronization etc. Given such specifications, it is desirable to synthesize a reference trajectory that is both optimal considering the dynamics of the vehicle and satisfies the temporal constraints. Motion planning [71], [1], at its early stage, considered optimal planning to reach a goal from an initial position while avoiding obstacles [72]. New techniques such as artificial potential functions [73], [74], cell decomposition and probabilistic roadmaps [1] are introduced for efficient planning in complex environment [75], [76] and high dimensional state-space. However, these approaches failed when task specifications have multiple goals or specific ordering of goals, for example surveying some areas in particular sequence.

Temporal logic [32–34] provides a compact mathematical formulation for specifying such complex mission specifications. Previous approaches mainly focus on the

usage of linear temporal logic (LTL), which can specify tasks such as visiting goals, periodically surveying areas, staying stable and safe. The main drawback of the LTL formulation is that it cannot specify time distance between tasks. In surveillance examples, a simple task may be to individually monitor multiple areas for at least x amount of time. Additionally, the LTL formulation commonly assumes the environment to be static. Traditional approaches commonly start with creating a finite abstraction of the environment including the dynamics, then combine it with the automata that is generated from the LTL specification [77]. The cell decomposition performed in the abstraction process requires the environment to be static; but in most situations this is not the case. For example, the use of Unmanned Aerial Vehicles (UAVs) for surveillance in the commercial airspace needs to consider motion of other aircraft. The other weakness of the automata-based approach is that it is computationally expensive. In this work, we are interested in motion planning for surveillance in an airspace with finite time task constraints and safety guarantees. For this work, we only consider the other aircraft in the target area as dynamic obstacles to the UAV. Further, we assume that the motions of these dynamic obstacles can be either predicted during the planning or are known a priori.

Due to the limitations of the previous approaches, we instead present a method based on metric temporal logic (MTL) [78], [79] and an optimization problem formulation to solve the planning problem. MTL extends the LTL [34] temporal operators so that it can express the requirements on time distance between events and event durations. This allows us to describe the dynamic obstacle and survey durations in our case study.

An optimization based method for LTL was previously proposed and extended by [68, 80]. In [80], the authors propose to transform LTL specifications to mixed integer constraints and solve the planning problem for finite horizon using either a mixed integer linear program (MILP) or a mixed integer quadratic program (MIQP) solver. In [68], the algorithm is extended to infinite horizon so that trajectories can contain loops. However, none of the methods consider dynamic environment or moving obstacles, time varying constraints, or duration of the tasks. MTL was used as a temporal constraint to a routing planning problem in [81]. Their formulation unfortunately does not allow users to incorporate dynamics of the vehicle.

In this chapter of the thesis, we consider a path planning problem for surveillance under survey durations constraints for each region and overall temporal constraint to visit each region within given times. Our problem is considerably different from the problems formulated in the existing literature in the sense that we not only consider dynamic environment but also associate each subtask with a duration constraint. We generate a path that guarantees safety by avoiding static and moving obstacles in the workspace and the path is optimal in the sense that it minimizes a predefined cost function. We do not adopt the linear encoding from [68], since moving obstacles is not periodic in nature. Similar to their approach, we adopt the usage of mixed logic dynamic (MLD) to model vehicle dynamics, so that the overall problem is a MILP (considering linear cost function).

Our main contribution is usage of MTL to specify time bounded tasks for the mission planner and reformulation of the problem into a MILP. We also demonstrate the methods on the case studies of using a quadrotor and ground vehicle to survey

multiple areas given the MTL specifications. The rest of this chapter is organized as follows. In section 4.2 we present the fundamentals of MTL and define the overall motion planning problem. We then formulate it into a mixed integer linear optimization problem incorporating the temporal constraints in section 4.3. Afterward, we demonstrate our approach on motion planning for different simulation setups.

4.2 Preliminaries

In this part we consider a surveying task in an area by a robot whose dynamics are given by the nonlinear model (4.1).

$$x(t+1) = f(t, x(t), u(t)) \quad (4.1)$$

where $x(t) \in \mathcal{X}$, $x(0) \in \mathcal{X}_0 \subseteq \mathcal{X}$, and $u(t) \in \mathcal{U}$ for all $t = 0, 1, 2, \dots$. Let us denote the trajectory of the system (4.1) starting at t_0 with initial condition x_0 and input $u(t)$ as $\mathbf{x}_{t_0}^{x_0, u} = \{x(s) \mid s \geq t_0, x(t+1) = f(t, x(t), u(t)), x(t_0) = x_0\}$. For brevity, we will use \mathbf{x}_{t_0} instead of $\mathbf{x}_{t_0}^{x_0, u}$ whenever we do not need the explicit information about $u(t)$ and x_0 .

Definition 4.2.1. *An atomic proposition is a statement about the system variables (x) that is either **True**(\top) or **False**(\perp) for some given values of the state variables.*
[80]

Let $\Pi = \{\pi_1, \pi_2, \dots, \pi_n\}$ be the set of atomic propositions which labels \mathcal{X} as a collection of survey areas, free space, obstacles etc. The moving obstacles make the free space to change from time to time, and hence labeling the environment is time

dependent. We define a map that labels the time varying environment as follows

$$L : \mathcal{X} \times \mathcal{I} \rightarrow 2^\Pi \quad (4.2)$$

where $\mathcal{I} = \{[a, b] \mid b > a \geq 0\}$ and 2^Π is denoted as the power set of Π . In general, \mathcal{I} is used to denote a time interval but it can be used to denote a time instance as well.

The trajectory of the system is a sequence of states such that each state $x(t)$ stays in \mathcal{X} for all t and there exists $u(t) \in \mathcal{U}$ for all t such that $x(t+1) = f(t, x(t), u(t))$. Corresponding to each trajectory \mathbf{x}_0 , the sequence of atomic proposition satisfied is given by $\mathcal{L}(\mathbf{x}_0) = L(x(0), 0)L(x(1), 1)\dots$

The high level specification of the surveying task will be expressed formally using MTL which can incorporate timing specifications.

4.2.1 Metric Temporal Logic (MTL)

Metric temporal logic, a member of temporal logic family, deals with model checking under timing constraints. The formulas for MTL are build on atomic propositions by obeying some grammar.

Definition 4.2.2. *The syntax of MTL formulas are defined according to the following grammar rules:*

$$\phi ::= \top \mid \pi \mid \neg\phi \mid \phi \vee \phi \mid \phi \mathbf{U}_I \phi$$

where $I \subseteq [0, \infty]$ is an interval with end points in $\mathbb{N} \cup \{\infty\}$. $\pi \in \Pi$, \top and $\perp (= \neg\top)$ are the Boolean constants *true* and *false* respectively. \vee denotes

the disjunction operator and \neg denotes the negation operator. \mathbf{U}_I symbolizes the timed Until operator. Sometimes we will represent $\mathbf{U}_{[0,\infty]}$ by \mathbf{U} . Other Boolean and temporal operators such as conjunction (\wedge), eventually within I (\diamond_I), always on I (\square_I) etc. can be represented using the grammar desired in definition 5.2.2. For example, we can express time constrained eventually operator $\diamond_I\phi \equiv \top\mathbf{U}_I\phi$ and so on.

Definition 4.2.3. *The semantics of any MTL formula ϕ is recursively defined over a trajectory x_t as:*

$$x_t \models \pi \text{ iff } \pi \in L(x(t), t)$$

$$x_t \models \neg\pi \text{ iff } \pi \notin L(x(t), t)$$

$$x_t \models \phi_1 \vee \phi_2 \text{ iff } x_t \models \phi_1 \text{ or } x_t \models \phi_2$$

$$x_t \models \phi_1 \wedge \phi_2 \text{ iff } x_t \models \phi_1 \text{ and } x_t \models \phi_2$$

$$x_t \models \bigcirc\phi \text{ iff } x_{t+1} \models \phi$$

$$x_t \models \phi_1\mathbf{U}_I\phi_2 \text{ iff } \exists s \in I \text{ s.t. } x_{t+s} \models \phi_2 \text{ and } \forall s' \leq s, x_{t+s'} \models \phi_1.$$

Thus, the expression $\phi_1\mathbf{U}_I\phi_2$ means that ϕ_2 will be true within time interval I and until ϕ_2 becomes true ϕ_1 must be true. Similarly, the release operator $\phi_1\mathbf{R}\phi_2$ denotes the specification that ϕ_2 should always hold and it can be released only when ϕ_1 is true. The MTL operator $\bigcirc\phi$ means that the specification ϕ is true at next time instance, $\square_I\phi$ means that ϕ is always true for the time duration I , $\diamond_I\phi$ means that ϕ will eventually become true within the time interval I . Composition of two or more MTL operators can express very sophisticated specifications; for example $\diamond_{I_1}\square_{I_2}\phi$ means that within time interval I_1 , ϕ will be true and from that

instance it will hold true always for a duration of I_2 . Other Boolean operators such as implication (\Rightarrow) and equivalence (\Leftrightarrow) can be expressed using the grammar rules and semantics given in definitions 4.2.2 and 4.2.3. More details on MTL grammar and semantics can be found in [78]. Satisfaction of a temporal specification ϕ by a trajectory \mathbf{x}_{t_0} will be denoted as $\mathbf{x}_{t_0} \models \phi$.

4.3 Problem Formulation and Solution

We consider a planning problem to periodically survey some selected areas in a given workspace. There are specific time bounds, associated with the regions, by which the surveillance has to be finished. Given the system dynamics (4.1), the objective is to find a suitable control law that will steer the robot in the survey area so that all regions are surveyed within the time bound and that control will optimize some cost function as well. The surveying task and its associated timing constraints and safety constraints can be expressed formally by Metric Temporal Logic (MTL). Let ϕ denote the MTL formula for the surveying task, and $J(x(t, u), u)$ be a cost function to make the path optimal in some sense. We formally present our planning objective as an optimization problem given in Problem (4.3.1)

Problem 4.3.1.

$$\begin{aligned} & \min_u && J(x(t, u), u(t)) \\ & \text{subject to} && x(t+1) = f(t, x(t), u(t)) \\ & && \mathbf{x}_{t_0} \models \phi \end{aligned}$$

In the following section we are going to discuss the linearization techniques for the dynamics of the robot, and our approach to translate an MTL constraint as

linear constraints.

4.3.1 Linearized Dynamics of the Robot

Since we are interested in solving the planning problem as an optimization problem with mixed integer and linear constraints, we need to represent the dynamics of the robot as a linear constraint to the optimization problem (4.3.1). We will consider surveying some given areas by ground robots as well as aerial robots. For quadrotor dynamics please refer to section 3.4.

4.3.1.1 Car-Like Model

For a car-like dynamical system (4.3), the system has three state variables: positions (x, y) , heading angle θ .

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad (4.3)$$

where u_1 and u_2 are the control inputs. We linearize this nonlinear model about several values for θ and depending on the value of θ , the closest linearization was used to drive the linearized system. The linearization is similar to what is suggested in [68]. The linearized system matrices at $\hat{\theta}$ are given by:

$$A = \begin{bmatrix} 0 & 0 & -\hat{u}_1 \sin(\hat{\theta}) \\ 0 & 0 & \hat{u}_1 \cos(\hat{\theta}) \\ 0 & 0 & 0 \end{bmatrix}; B = \begin{bmatrix} \cos(\hat{\theta}) & 0 \\ \sin(\hat{\theta}) & 0 \\ 0 & 1 \end{bmatrix}$$

4.3.2 Mixed Integer Linear Constraints

In this section, we demonstrate our approach to translate a time-bounded temporal logic formula to linear constraints on state variables and inputs. The easiest example would be how to express the temporal constraint that $x(t)$ lies within a convex polygon \mathcal{P} at time t . This simple example will serve as a building block for other complicated temporal operators. Any convex polygon \mathcal{P} can be represented as an intersection of several halfspaces. A halfspace is expressed by a set of points, $\mathcal{H}_i = \{x : h_i^T x \leq k_i\}$. Thus, $x(t) \in \mathcal{P}$ is equivalent to $x(t) \in \bigcap_{i=1}^n \mathcal{H}_i = \bigcap_{i=1}^n \{x : h_i^T x \leq k_i\}$. The temporal constraint that $x(t)$ will be inside \mathcal{P} for all $t \in \{t_1, t_1 + 1, \dots, t_1 + n\}$ can be represented by the set of linear constraints $\{h_i^T x(t) \leq k_i\}$ for all $i = \{1, 2, \dots, n\}$ and $\forall t \in \{t_1, t_1 + 1, \dots, t_1 + n\}$.

We adopted a method similar to the method used in [80] to translate temporal constraint ϕ into mixed integer linear constraints. We extend it to incorporate duration for task completion and loop constraints of the trajectory. Comparing to [68], we only enforce the loop constraints at the trajectory level instead of the temporal logic level, since the moving obstacles is not periodic in nature. The planning process will be repeated when the vehicle returns to the initial point.

In a polygonal environment, atomic propositions (AP), $p \in \Pi$, can be related to states of the system using disjunction and conjunction of halfspaces. In other words, the relationship between measured outputs such as location of the vehicle and the halfspaces defines the proposition used in the temporal logics. Consider the convex polygon case and let $z_i^t \in \{0, 1\}$ be the binary variables associated with

halfspaces $\{x(t) : h_i^T x(t) \leq k_i\}$ at time $t = 0, \dots, N$. We enforce the following constraint $z_i^t = 1$ if and only if $h_i^T x(t) \leq k_i$ by adding the linear constraints,

$$h_i^T x(t) \leq k_i + M(1 - z_i^t) \quad (4.4)$$

$$h_i^T x(t) \geq k_i - Mz_i^t + \epsilon$$

where M is a large positive number and ϵ is a small positive number. If we denote $P_t^{\mathcal{P}} = \bigwedge_{i=1}^n z_i^t$, then $P_t^{\mathcal{P}} = 1$ if and only if $x(t) \in \mathcal{P}$. This can be extended to the nonconvex case by decomposing the polygon to convex ones and linking them using disjunction operators. As discussed later in this section, the disjunction operator can also be translated to mixed integer linear constraints.

We will use $F_\phi(x, z, u, t)$ to denote the set of all mixed integer linear constraints corresponding to the temporal logic formula ϕ . Once we have formulated $F_p(x, z, u, t)$ for atomic propositions p , we can find $F_\phi(x, z, u, t)$ for any MTL formula ϕ . The next essential part of the semantics of MTL is the Boolean operations, such as \neg , \wedge , \vee . Similarly these operators can be translated into linear constraints. Let $t \in \{0, 1, \dots, N\}$, P_t^ϕ be the continuous variables within $[0,1]$ associated with formula ϕ made up with propositions $p \in \Pi$ at time t .

- $\phi = \neg p$ is the negation of an atomic proposition, and it can be modeled as

$$P_t^\phi = 1 - P_t^p. \quad (4.5)$$

- The conjunction operation, $\phi = \bigwedge_{i=1}^m p_i$, is modeled as

$$\begin{aligned} P_t^\phi &\leq P_t^{p_i}, \quad i = 1, \dots, m, \\ P_t^\phi &\geq 1 - m + \sum_{i=1}^m P_t^{p_i}, \end{aligned} \quad (4.6)$$

- The disjunction operator, $\phi = \bigvee_{i=1}^m p_i$, is modeled as

$$\begin{aligned} P_t^\phi &\geq P_t^{p_i}, \quad i = 1, \dots, m, \\ P_t^\phi &\leq \sum_{i=1}^m P_t^{p_i}, \end{aligned} \tag{4.7}$$

Similarly, the temporal operators can be modeled using linear constraints as well. Let $t \in \{0, 1, \dots, N - t_2\}$, where $[t_1, t_2]$ is the time interval used in the MTL.

- Eventually: $\phi = \diamond_{[t_1, t_2]} p$ is equivalent to

$$\begin{aligned} P_t^\phi &\geq P_\tau^p, \quad \tau \in \{t + t_1, \dots, t + t_2\} \\ P_t^\phi &\leq \sum_{\tau=t+t_1}^{t+t_2} P_\tau^p \end{aligned} \tag{4.8}$$

- Always: $\phi = \square_{[t_1, t_2]} p$ is equivalent to

$$\begin{aligned} P_t^\phi &\leq P_\tau^p, \quad \tau \in \{t + t_1, \dots, t + t_2\} \\ P_t^\phi &\geq \sum_{\tau=t+t_1}^{t+t_2} P_\tau^p - (t_2 - t_1), \end{aligned} \tag{4.9}$$

- Until: $\phi = p\mathcal{U}_{[t_1, t_2]} q$ is equivalent to

$$\begin{aligned} a_{tj} &\leq P_q^j \quad j \in \{t + t_1, \dots, t + t_2\}, \\ a_{tj} &\leq P_p^k \quad k \in \{t, \dots, j - 1\}, j \in \{t + t_1, \dots, t + t_2\}, \\ a_{tj} &\geq P_q^j + \sum_{k=t}^{j-1} P_p^k - (j - t) \quad j \in \{t + t_1, \dots, t + t_2\}, \\ P_t^\phi &\leq \sum_{j=t+t_1}^{t+t_2} a_{tj}, \\ P_t^\phi &\geq a_{tj} \quad j \in \{t + t_1, \dots, t + t_2\}. \end{aligned} \tag{4.10}$$

The until formulation (4.10) is obtained similarly to [80]. For MTL, it is modified by noticing the following equality,

$$P_t^\phi = \bigvee_{j=t+t_1}^{t+t_2} \left(\left(\bigwedge_{k=t}^{k=j-1} P_p^k \right) \wedge P_q^j \right).$$

Other combinations for different temporal operators are also straight forward and we are not enumerating them for the sake of space, but one can easily derive them by using (4.5) through (4.10).

Using this approach, we translate the given high level specification in MTL $(\mathbf{x}_{t_0} \models \phi)$ to a set of mixed integer linear constraints $F_\phi(x, z, u, t)$. At the end, we add the constraint $P_0^\phi = 1$, i.e. the overall specification ϕ is satisfied. Since Boolean variables are only introduced when halfspaces are defined, the computation cost of MILP is at most exponential to the number of halfspaces times the discrete steps N .

4.4 Case Study and Discussion

We apply our method for solving mission planning with finite time constraints on two different workspaces. Both workspaces contain static and moving obstacles. The experiments are run through YALMIP-CPLEX on a computer with 3.4GHz processor and 8GB memory. We performed the simulation for both a quadrotor model and a car model.

The first environment is the one shown in Fig. 4.1, where blue and gray areas represent moving and static obstacles respectively, and green areas represent survey targets.

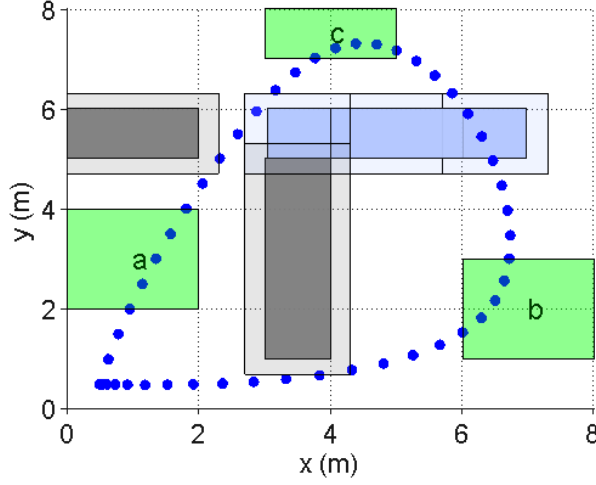


Figure 4.1: Workspace setup of the first test case. Blue area represents an obstacle moving from right to left. Grey areas represent static obstacles. Green areas represent survey areas used in the temporal logic. The shaded areas around obstacles represent boundary of the obstacles for discrete planning, so that the obstacles can be avoided even for continuous dynamics. The resulting 2D trajectory of the quadrotor for ϕ_1 is shown as blue dots. The resulting motion is counter clockwise.

Let the temporal specification be the following

$$\phi_1 = \diamond \square_{[0,2]} A \wedge \diamond \square_{[0,2]} B \wedge \diamond \square_{[0,2]} C \wedge \square \neg O$$

$$\phi_2 = \diamond \square_{[0,2]} A \wedge \diamond \square_{[0,2]} B \wedge \diamond \square_{[0,2]} C \wedge \square \neg O \wedge \neg B \mathcal{U}_{[0,N]} A$$

where O represents the static and moving obstacles, N is the horizon of the planning trajectory. Such N can be generally obtained by performing a feasibility test using MILP solver starting from $N = 2$, and increasing N until finding a feasible one. For the purpose of comparing different temporal constraints, we choose a feasible horizon $N = 50$ for both MTL specifications ϕ_1 and ϕ_2 . The specification ϕ_1 requires the vehicle to visit areas A , B and C eventually and stay there for at least 2 time units, while avoiding obstacles O . ϕ_2 adds an additional requirement on the ordering between A and B , so that region A has to be visited first. We consider the

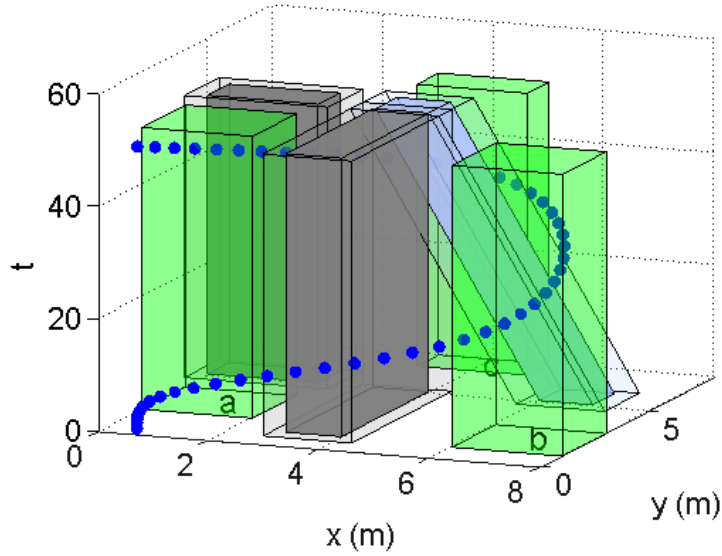


Figure 4.2: Time-state-space representation of the environment and resulting trajectory for the quadrotor with temporal constraints ϕ_1 . The vehicle starts from (0.5,0.5) and surveys area b, c and a sequentially.

cost function, J , to be minimized is $\sum_{t=0}^N |u(t)|$. The dynamics of the vehicles are discretized at a rate of 2Hz.

The resulting trajectory for the quadrotor with temporal specification ϕ_1 is plotted in time-state-space in Fig. 4.2. The projection of the trajectory on the workspace is shown in Fig. 4.1. The motion of the quadrotor in Fig. 4.1 is counter clockwise. The quadrotor safely avoids the moving obstacle by navigating through the area after the obstacle passes. The survey duration in individual area also satisfies the requirements as shown in Fig. 4.1. The quadrotor stays within each survey area for 5 discrete time units which is equivalent to 2s duration. These plots show a better realization of visiting individual areas comparing to [81] and [68]. In their results, trajectory visits targeted areas for only 1 discrete time unit which is not desired for most surveillance tasks. Additional local planning is possible to

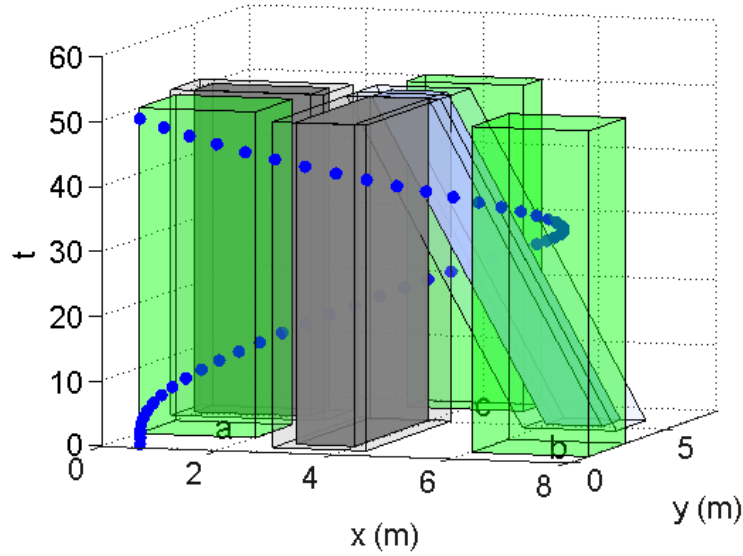


Figure 4.3: Time-state-space representation of the environment and resulting trajectory for the quadrotor with temporal constraints ϕ_2 . Because of the additional ordering requirement, the vehicle covers area a first before visiting area b.

generate a sweeping pattern so that the target areas can be covered by onboard sensors in designed durations.

The resulting trajectory for the vehicle with specification ϕ_2 is shown in time-state-space in Fig. 4.3. As can be verified from the trajectory, the vehicle travels first to area A before visiting B as specified. The CPLEX solver returns the solution for the first trajectory in 20.8 sec while the second one takes 34.7 sec. The additional continuous variables used in the until encoding have large influence in the performance. One of the possible future research directions would be finding different encoding of the until operator to improve the speed of the algorithm.

The result of specification ϕ_1 for the car model is shown in Fig. 4.4, where the small blue arrow associated with each blue dot indicates the instantaneous heading of the vehicle. The computation time is 150s. The longer computation time is caused

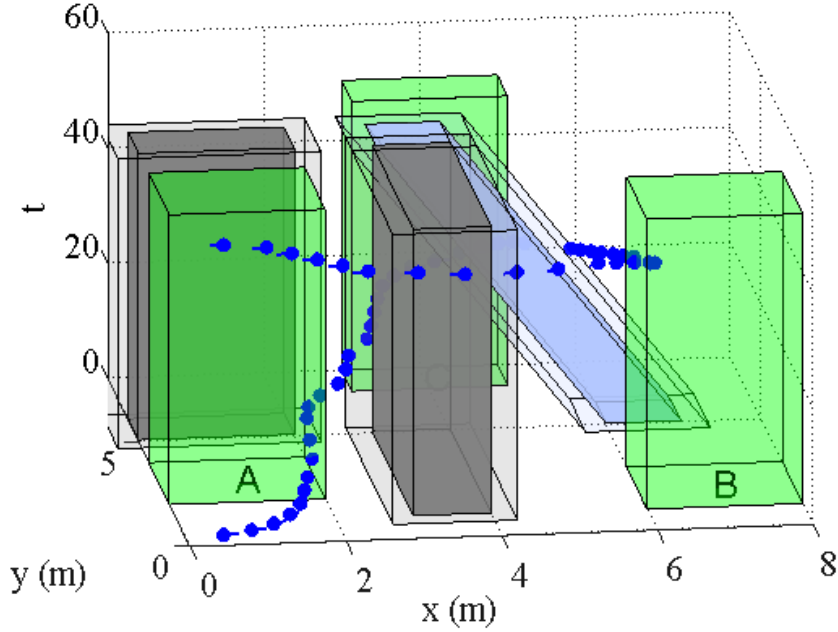


Figure 4.4: Time-state-space representation of the environment and resulting trajectory for the car model with temporal constraints ϕ_1 . The result for ϕ_2 is similar since the optimal solution for ϕ_1 already goes through area A first.

by the additional binary variables introduced by the linearization of the dynamics at various heading angles.

The second environment considers a fast moving obstacle that moves across the workspace diagonally, and hence the vehicle has to adjust its motion accordingly. The environment is shown in Fig 4.5. Similar to the previous example, it shows the motion of the moving obstacle, static obstacles and survey areas.

The temporal logic specifications are similar to the previous one but have an additional area to be visited. We also tested the case when certain area has to be visited first. The result is similar to previous cases, so we only show the plots for ϕ_3 .

$$\phi_3 = \diamond \square_{[0,2]} A \wedge \diamond \square_{[0,2]} B \wedge \diamond \square_{[0,2]} C \wedge \diamond \square_{[0,2]} D \wedge \square \neg O$$

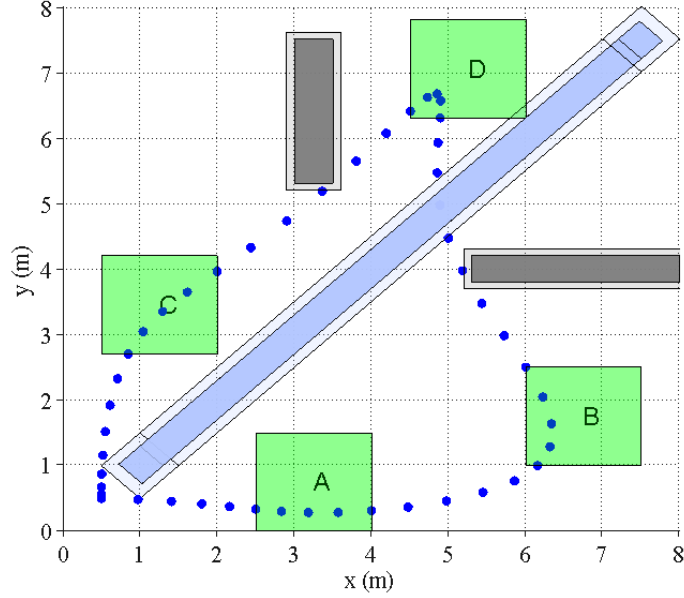


Figure 4.5: Workspace setup of the second test case. The moving obstacle moves from the the bottom left corner to upper right corner. 2D trajectory of the vehicle is shown as blue dots. The resulting motion is clockwise.

$$\phi_4 = \diamond \square_{[0,2]}A \wedge \diamond \square_{[0,2]}B \wedge \diamond \square_{[0,2]}C \wedge \diamond \square_{[0,2]}D \wedge \square \neg O \wedge \neg C \mathcal{U}_{[0,N]}A$$

The resulting trajectory in time-state-space for the quadrotor with temporal specification ϕ_3 is plotted in Fig. 4.6. The projected trajectory on the workspace of the robot is shown in Fig. 4.5. The motion of the vehicle is clockwise in Fig. 4.5. As can be seen from Fig. 4.6, the quadrotor safely avoids the moving obstacle and the static ones nearby. The survey duration in individual area also satisfies the requirements as shown in Fig. 4.5. The computation time is 138.8s. The increase in computation time is because the complex environment introduces more binary variables. Similarly the resulting trajectory for the quadrotor with temporal specification ϕ_4 is shown in Fig. 4.7. The vehicle travels first to area *A* as specified.

The result of specification ϕ_3 for the car model is shown in Fig. 4.8, where the blue arrows indicate the heading of the vehicle. The computation time is 500s.

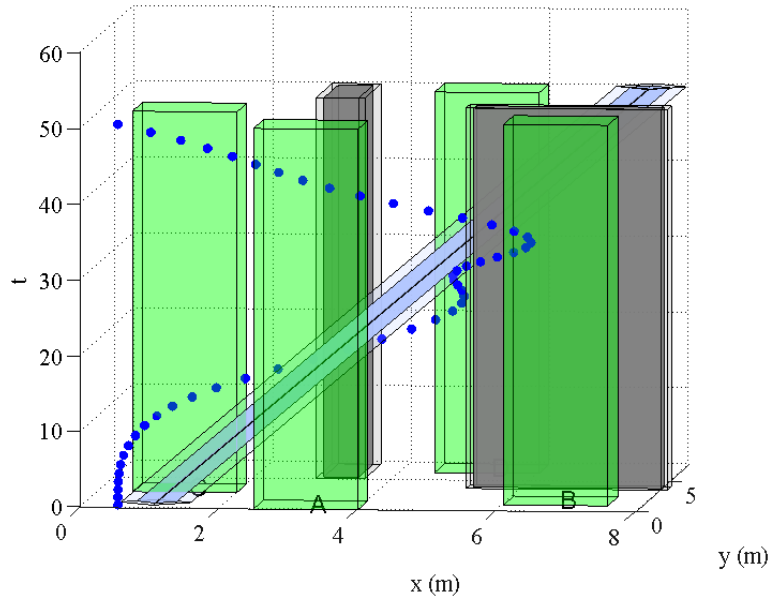


Figure 4.6: Time-state-space representation of the environment and resulting trajectory for the quadrotor with temporal constraints ϕ_3 . The planned trajectory is very close to the dynamic obstacles.

4.5 Conclusion

In this chapter, we have presented an optimization based approach to plan the trajectory of a robot in a dynamic environment to perform some temporal task with finite time constraints. Our approach is simple in the sense that it translates the time constraints and the temporal specifications as linear constraints on the state and input variables. We have linearized the dynamics of the robot in order to formulate the problem as a Linear Programming problem. We have considered polygonal environments for our case studies, but if the environment is not polygonal, one can approximate it with a polygonal environment. We have used a binary variable (z) with each halfspace, so if the polygonal approximation of the environment contains too many halfspaces, the problem would be complex.

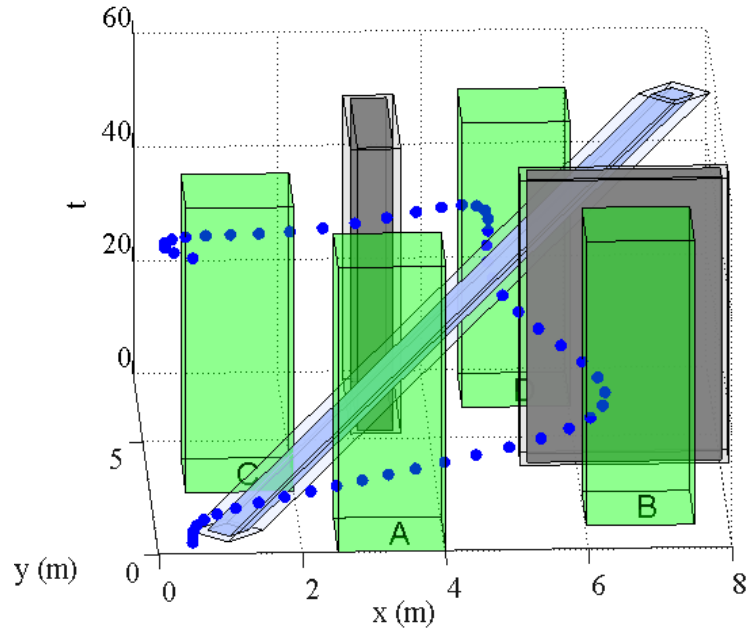


Figure 4.7: Time-state-space representation of the environment and resulting trajectory for the quadrotor with temporal constraints ϕ_4 . Because of the additional ordering requirement, the vehicle covers area A first before visiting area C.

We have reported the case studies for quadrotor and car dynamics. The simulation results show promising performance of our approach to find an optimal solution. We consider dynamic but deterministic environments and uncertainties in the dynamics of the robot are also not considered. There are many possible directions such as planning in uncertain environment, stochastic dynamics of the robots or multi-robot cooperative planning that one might consider as an extension of this framework.

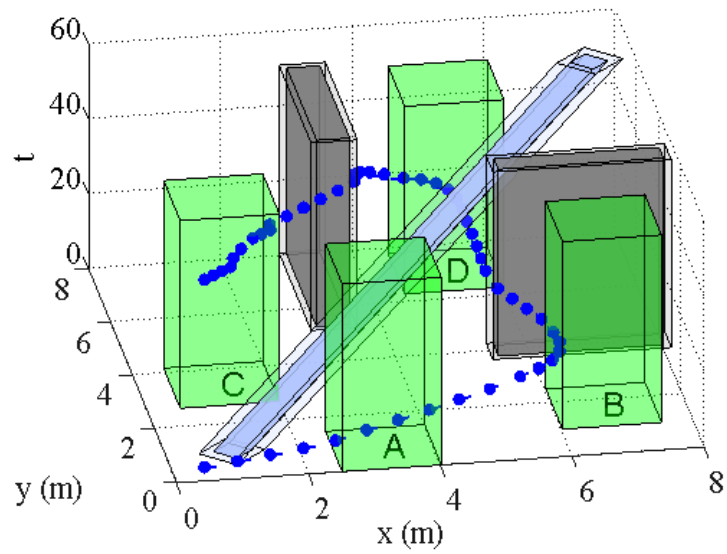


Figure 4.8: Time-state-space representation of the environment and resulting trajectory for the car model with temporal constraints ϕ_3 . The result for ϕ_4 is the same since the optimal solution already goes through A first.

Chapter 5: Timed Automata Based Motion Planning

5.1 Overview

Motion planning and task planning have gained an enormous thrust in the robotics community in the past decade or so. Although motion (task) planning has attracted a great deal of research in the past few decades, recently, researchers have come up with new metrics and methodologies to represent motion and task specifications. Initially, motion planning for a mobile robot started with the aim of moving a point mass from an initial position to a final position in some optimal fashion. Through the course of time, people started to consider planning in cluttered domains (i.e. in presence of obstacles) and also accounted for the dimensionality and the physical constraints of the robot.

Though we have efficient approaches for general motion planning, very few are available or scalable to plan in dynamic environments or under finite time constraints. Temporal logics have been used greatly to address complex motion specifications, motion sequencing and timing behaviors etc. Historically temporal logic was originated for model checking, validation and verification in the software community [34] and later on researchers found it very helpful to use Linear Temporal logic (LTL), Computational Tree logic (CTL), Signal Temporal logic (STL) etc. for

representing complex motion (or task) specifications. The developments of tools such as SPIN [35], NuSMV [36] made it easier to check if given specifications can be met by creating a suitable automaton and looking for a feasible path for that automaton. However, the construction of the automaton from a given temporal logic formula is based on the implicit assumption that there is no time constraints associated with the specification.

Currently motion planning for robots is in such a stage where it is very crucial to incorporate time constraints since these constraints can arise from different aspects of the problem: dynamic environment, sequential processing, time optimality etc. Planning with bounded time objectives is inherently hard due to the fact that every transition from one state to another in the associated automaton has to be carried out, by some controller, exactly in time from an initial configuration to the final configuration. Bounded time motion planning has been done in heuristic ways [37, 38] and also by using a mixed integer linear programming (MILP) framework [82, 83]. In this chapter, we are interested in extending the idea of using LTL for time-unconstrained planning to use MITL for time-constrained motion planning. In [84], the authors proposed a method to represent time constrained planning task as an LTL formula rather than MITL formula. This formulation reduced the complexity of EXP-SPACE-complete for MITL to PSPACE-complete for LTL. However, the number of states in the generated Büchi automata increases with the numbers of steps.

In this work, we mainly focus on motion planning based on the construction of an efficient timed automaton from a given MITL specification. A dedicated

controller to navigate the robot can be constructed for the general planning problem once the discrete path is obtained from the automaton. The earlier results on construction of algorithms to verify timing properties of real time systems can be found in [85]. The complexity of satisfiability and model checking problems for MTL formulas has been already studied in [79] and it has been shown that commonly used real-time properties such as bounded response and invariance can be decided in polynomial time or exponential space. More works on the decidability on MTL can be found in [86] and the references there in. The concept of alternating timed automata for bounded time model checking can be found in [87]. [88] describes the construction of deterministic timed automata from MTL specifications and this provides a unified framework to include all the future operators of MTL. The key to the approach of [88] was in separating the continuous time monitoring from the discrete time predictions of the future. We restrict our attention to generate timed automata from MITL based on the work done in [89]. This is accomplished by constructing a timed automaton to generate a sequence of states and another one to check whether the sequence generated is actually a valid one in the sense that it satisfies the given MITL specification.

The rest of the chapter is organized as follows, section 5.2 provides a background on MITL and the timed automata based approach for MITL. Section 5.3 illustrates how the timed automata can be used for motion synthesis and we also provide UPPAAL [90] implementation of the same. Section 5.4 gives some examples on different bounded time tasks and shows the implementation results. Section 5.5 provides a brief overview of how a continuous trajectory can be generated from the

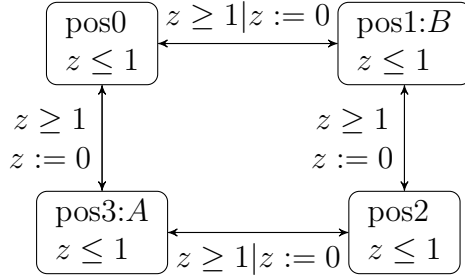


Figure 5.1: Timed Automata based on cell decomposition and robot dynamics

discrete plan. Finally we conclude in section 5.6. This part of the work is published in [91]

5.2 Preliminaries

In this chapter, we consider a surveying task in an area by a robot whose motion is abstracted to a graph. In particular for our particular setup, the robot motion is captured as a timed automaton (Fig. 5.1). Every edge is a timed transition that represents navigation of the robot from one location to another in space and every vertex of the graph represents a partition of the space. Our objective is to find an optimal time path that satisfies the specifications given by timed temporal logic.

5.2.1 Metric Interval Temporal Logic (MITL)

Metric interval temporal logic is a specification that includes timed temporal specifications for model checking. It differs from Linear Temporal Logic in the part that it has bounded time constraints on the temporal operators.

The formulas for LTL are build on atomic propositions by obeying the following

grammar.

Definition 5.2.1. *The syntax of LTL formulas are defined according to the following grammar rules:*

$$\phi ::= \top \mid \pi \mid \neg\phi \mid \phi \vee \phi \mid \mathbf{X}\phi \mid \phi \mathbf{U}\phi$$

$\pi \in \Pi$ the set of propositions, \top and $\perp (= \neg\top)$ are the Boolean constants *true* and *false* respectively. \vee denotes the disjunction operator and \neg denotes the negation operator. \mathbf{U} represents the Until operator. MITL extends the Until operator to incorporate timing constraints.

Definition 5.2.2. *The syntax of MITL formulas are defined according to the following grammar rules:*

$$\phi ::= \top \mid \pi \mid \neg\phi \mid \phi \vee \phi \mid \phi \mathbf{U}_I \phi$$

where $I \subseteq [0, \infty]$ is an interval with end points in $\mathbb{N} \cup \{\infty\}$. \mathbf{U}_I symbolizes the timed Until operator. Sometimes we will represent $\mathbf{U}_{[0, \infty]}$ by \mathbf{U} . Other Boolean and temporal operators such as conjunction (\wedge), eventually within I (\diamond_I), always on I (\square_I) etc. can be represented using the grammar described in definition 5.2.2. For example, we can express time constrained eventually operator $\diamond_I \phi \equiv \top \mathbf{U}_I \phi$ and so on. In this chapter all the untimed temporal logic specifications are transformed into expressions involving the Until operator and all the timed operator specifications are transformed to eventually within I , to make it easier to generate a timed automaton.

MITL is interpreted over n -dimensional Boolean ω -sequences of the form $\xi : \mathbb{N} \rightarrow \mathbb{B}^n$, where n is the number of propositions.

Definition 5.2.3. *The semantics of any MTL formula ϕ is recursively defined over a trajectory (ξ, t) as:*

$(\xi, t) \models \pi$ iff (ξ, t) satisfies π at time t

$(\xi, t) \models \neg\pi$ iff (ξ, t) does not satisfy π at time t

$(\xi, t) \models \phi_1 \vee \phi_2$ iff $(\xi, t) \models \phi_1$ or $(\xi, t) \models \phi_2$

$(\xi, t) \models \phi_1 \wedge \phi_2$ iff $(\xi, t) \models \phi_1$ and $(\xi, t) \models \phi_2$

$(\xi, t) \models \bigcirc\phi$ iff $(\xi, t + 1) \models \phi$

$(\xi, t) \models \phi_1 \mathbf{U}_I \phi_2$ iff $\exists s \in I$ s.t. $(\xi, t + s) \models \phi_2$ and $\forall s' \leq s, (\xi, t + s') \models \phi_1$.

Thus, the expression $\phi_1 \mathbf{U}_I \phi_2$ means that ϕ_2 will be true within time interval I and until ϕ_2 becomes true ϕ_1 must be true. The MITL operator $\bigcirc\phi$ means that the specification ϕ is true at next time instance, $\square_I\phi$ means that ϕ is always true for the time duration I , $\diamond_I\phi$ means that ϕ will eventually become true within the time interval I . Composition of two or more MITL operators can express very sophisticated specifications; for example $\diamond_{I_1}\square_{I_2}\phi$ means that within time interval I_1 , ϕ will be true and from that instance it will hold true always for a duration of I_2 . Other Boolean operators such as implication (\Rightarrow) and equivalence (\Leftrightarrow) can be expressed using the grammar rules and semantics given in definitions 5.2.2 and 5.2.3. More details on MITL grammar and semantics can be found in [78], [85].

5.2.2 MITL and timed automata based approach

An LTL formula can be transformed into a Büchi automaton which can be used in optimal path synthesis [92] and automata based guidance [93]. Similarly, in

this work, we focus on developing a timed automata based approach for MITL based motion planning. MITL, a modification of Metric Temporal Logics (MTL), disallows the punctuation in the temporal interval, so that the left boundary and the right boundary have to be different. In general the complexity of model checking for MTL related logic is higher than that of LTL. The theoretical model checking complexity for LTL is PSPACE-complete [94]. The algorithm that has been implemented is exponential to the size of the formula. MTL by itself is undecidable. The model checking process of MITL includes transforming it into a timed automaton [85] [89]. CoFlatMTL and BoundedMTL defined in [95] are more expressive fragments of MTL than MITL, which can be translated to LTL-Past but with exponential increase in size. SafetyMTL [86] and MTL, evaluated over finite and discrete timed words, can be translated into alternative timed automata. Although theoretically, the results suggest many fragments of MTL are usable, many algorithms developed for model checking are based on language emptiness check, which are very different from the control synthesis i.e. finding a feasible path. To the best of our knowledge, the algorithm that is closer to implementation for motion planning is that of [89].

This work uses the MITL and timed automaton generation based on [89]. In the following section, the summary of the transformation and our implementation for control synthesis are discussed.

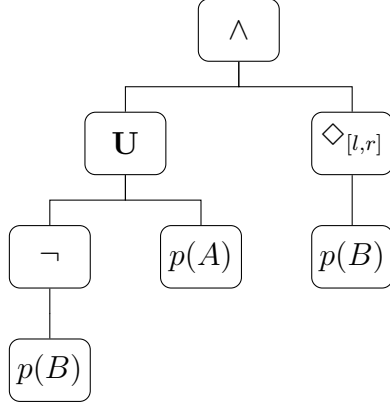


Figure 5.2: Logic tree representation of ϕ .

5.3 MITL for motion planning

5.3.1 MITL to Timed Automata Transformation

Consider the following requirements: a robot has to eventually visit an area A and another area B in time interval $[l, r]$, and the area A has to be visited first. This can be captured in the following MITL expression,

$$\phi = (\neg BUA) \wedge (\diamond_{[l,r]} B)$$

This expression can be represented by a logic tree structure, where every node that has children is a temporal logic operator and every leaf node is an atomic proposition, as shown in Fig. 5.2. Every link represents an input output relationship.

The authors in [89] propose to change every temporal logic operator into a timed signal transducer, which is a temporal automaton that accepts input and generates output. Based on the definitions in [89] the Timed Transducer (TT) used in this paper is defined in Definition 5.3.1 below, to fit the control synthesis problem,

Definition 5.3.1 (Timed Transducer). *An timed transducer is a tuple*

$$\mathcal{A} = (\Sigma, Q, \Gamma, \mathcal{C}, \lambda, \gamma, I, \Delta, q_0, F),$$

where

- Σ is the input alphabet, Q is the finite set of discrete states,
- Γ is the output alphabet, \mathcal{C} is the set of clock variables,
- I is the invariant condition defined by conjunction of inequalities of clock variables. The clock variables can be disabled and activated by setting the rate of the clock 0 or 1 in the invariant I .
- $\lambda : Q \rightarrow \Sigma$ is the input function, which labels every state to an input, while $\gamma : Q \rightarrow \Gamma$ is the output function, which labels every state to an output.
- Δ is the transition relationship between states which is defined by (p, q, r, g) , where p is the start state, q is the end state, r is the clock resets, and g is the guard condition on the clock variables.
- q_0 is the initial state of the timed automaton,
- F is the set of Büchi states that have to be visited infinitely often.

The transformation of Until operator and timed Eventually operator is summarized in Figs. 5.3, 5.4 and 5.5. This is based on [89] with minor changes to match with our definition of TT. In Fig. 5.3, the timed automaton for $p\mathcal{U}q$ is shown. The inputs outputs of the states are specified in the second line within the box of each

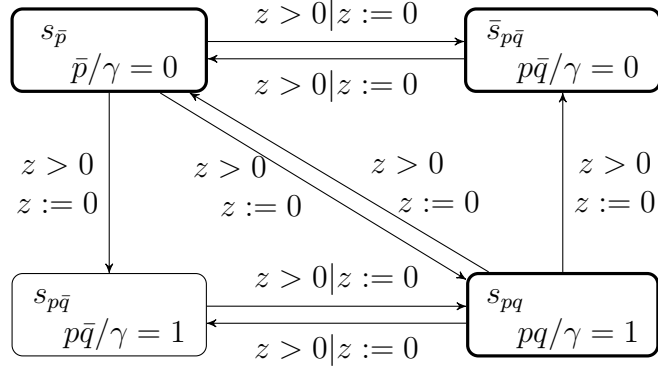


Figure 5.3: The timed automaton for $p\mathcal{U}q$. The inputs and outputs of the states are specified in the second line of each state. $p\bar{q}$ means the inputs are $[1, 0]$ and \bar{p} means the inputs can be $[0, 1]$ or $[0, 0]$, and $\gamma = 1$ means the output is 1. Transitions are specified in the format of guard|reset. In this case all the transitions have guard $z > 0$ and reset clock z . All states in this automaton are Büchi accepting states except $s_{p\bar{q}}$. The Büchi accepting states are highlighted.

state. $p\bar{q}$ means the inputs are $[1, 0]$ and \bar{p} means the inputs can be $[0, 1]$ or $[0, 0]$, and $\gamma = 1$ means the output is 1. Transitions are specified in the format of $g|r$. In this case, all the transitions have guard $z > 0$ and reset clock z . All states in this automaton are Büchi accepting states except $s_{p\bar{q}}$. The Büchi accepting states are highlighted.

The TT for timed eventually ($\diamond_I a$) is decomposed into two automata, the generator generates predictions of the future outputs of the system, while the checker verifies that the generated outputs actually fit the inputs. Detailed derivations and verifications of the models can be found in [89]. The composition between them is achieved through the shared clock variables. Additional synchronization (‘ch!’) is added in our case to determine the final satisfaction condition for the control synthesis. A finite time trajectory satisfies the MITL, when the output signal of the generator automaton (Fig. 5.4) includes a pair of raising edge and falling edge

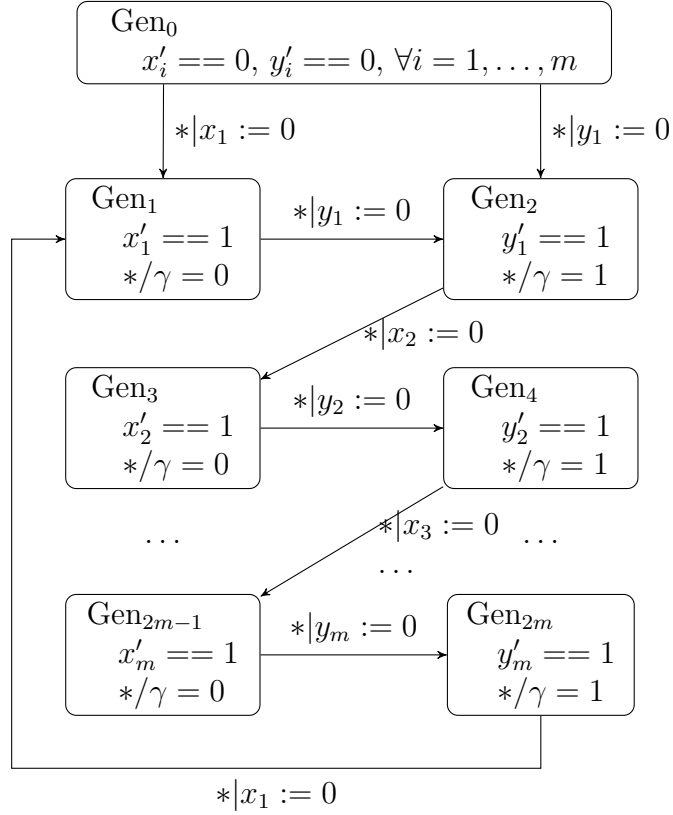


Figure 5.4: The timed automaton for the generator part of $\diamond_I a$ for motion planning. $2m$ is the number of clocks required to store the states of the timed eventually (\diamond_I) operator. It is computed based on the interval I . Detail computation and derivation can be found in [89]. x'_i represents the rate of the clock x_i . By setting the rate to be 0, we essentially deactivate the clock. The ‘*’ symbol means that there is no value for that particular input, output or guard for that state. There are no Büchi states since all the time is bounded

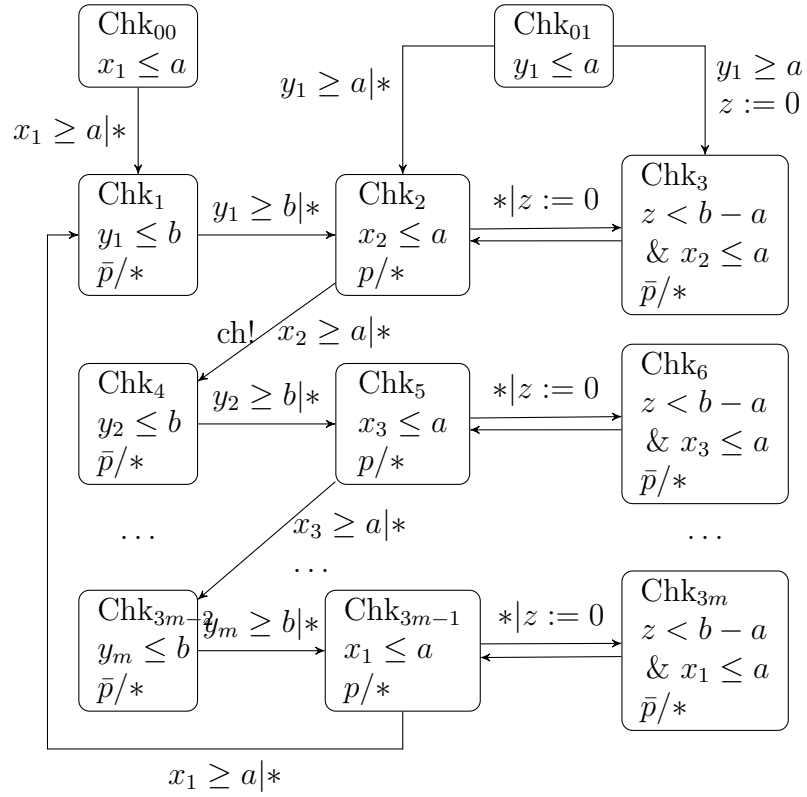


Figure 5.5: The timed automaton for the checker part of $\diamond_I a$ for motion planning. $2m$ is the number of clocks required for the timed eventually (\diamond_I) operator. There are no Büchi states since all the time is bounded

verified by the checker automaton. The transition from Chk_2 to Chk_4 (Fig. 5.5) marks the exact time when such falling edge is verified. This guarantees that the time trajectory before the synchronization is a finite time trajectory that satisfies the MITL.

The composition of TT based on logic trees such as that of Fig. 5.2 is defined similar to [89] with some modifications to handle cases when logic nodes have two children, for example the until and conjunction operators.

Definition 5.3.2 (I/O Composition).

Let $\mathcal{A}_1^1 = (\Sigma_1^1, Q_1^1, \Gamma_1^1, \mathcal{C}_1^1, \lambda_1^1, \gamma_1^1, I_1^1, \Delta_1^1, q_{10}^1, F_1^1)$, $\mathcal{A}_2^1 = (\Sigma_2^1, Q_2^1, \Gamma_2^1, \mathcal{C}_2^1, \lambda_2^1, \gamma_2^1, I_2^1, \Delta_2^1, q_{20}^1, F_2^1)$ be the input side of the automaton. If there is only one, then \mathcal{A}_1^1 is used. Let $\mathcal{A}^2 = (\Sigma^2, Q^2, \Gamma^2, \mathcal{C}^2, \lambda^2, \gamma^2, I^2, \Delta^2, q_0^2, F^2)$ be the output side of the automaton. Because of the input output relationship between them, they should satisfies the condition that $[\Gamma_1^1, \Gamma_2^1] = \Sigma^2$. The composition is an new TT such that,

$$\mathcal{A} = (\mathcal{A}_1^1, \mathcal{A}_2^1) \otimes (\mathcal{A}^2) = ([\Sigma_1^1, \Sigma_2^1], Q, \Gamma^2, \mathcal{C}, \lambda, \gamma, I, \Delta, q_0, F)$$

where

$$Q = \{(q_1^1, q_2^1, q^2) \in Q_1^1 \times Q_2^1 \times Q^2, \text{ s.t. } (\gamma_1^1(q_1^1), \gamma_2^1(q_2^1)) = \lambda^2(q^2)\}$$

$$\mathcal{C} = (\mathcal{C}_1^1 \cup \mathcal{C}_2^1 \cup \mathcal{C}^2), \lambda(q_1^1, q_2^1, q^2) = [\lambda_1^1(q_1^1), \lambda_2^1(q_2^1)], I_{(q_1^1, q_2^1, q^2)} = I_{(q_1^1, q_2^1)}^1 \cap I_{q^2}^2, q_0 = (q_{10}^1, q_{20}^1, q_0^2) \text{ and } F = F_1^1 \cap F_2^1 \cup F^2.$$

5.3.2 Path Synthesis using UPPAAL

The overall path synthesis framework is summarized as follows,

- First, robot and environments are abstracted to a timed automata \mathcal{T}_{map} using cell decomposition, and the time to navigate from one cell to another is estimated based on the robot's dynamics. For example Fig. 5.1.
- Second, MITL formula is translated to TT \mathcal{A} using the method described in the previous section.
- Then the product is taken of TT \mathcal{A} with the TA \mathcal{T}_{map} using the location label. For instance in Fig. 5.1 the product of $\text{pos1} : B$ will be taken with all states in TT that do not satisfy the predicate $p(a)$ but satisfy $p(b)$.
- The resulting timed automata are then automatically transformed to an UPPAAL [90] model with an additional satisfaction condition verifier. An initial state is chosen so that the output at that state is 1. Any finite trajectory which initiated from that state and satisfying the following conditions will satisfy the MITL specification. Firstly, it has to visit at least one of the Büchi accepting states, and secondly, it has to meet the acceptance condition for the timed eventually operator. To perform such a search in UPPAAL, a final state is added to allow transitions from any Büchi accepting state to itself. A verification automaton is created to check the finite acceptance conditions for every timed eventually operator.
- An optimal timed path is then synthesized using the UPPAAL verification tool.

The implementation of the first and the second step is based on parsing and simplification functions of `ltl2ba` tool [96] with additional capabilities to generate TT. We then use the generated TT to autogenerate a PYTHON script which constructs the UPPAAL model automatically through `PyUPPAAL`, a PYTHON interface to create and layout models for UPPAAL. The complete set of tools¹ is implemented in C to optimize the speed.

5.4 Case Study and Discussion

We demonstrate our framework for a simple environment and for some typical temporal logic formulas. Although our tool is not limited by the complexity of the environment, we use a simple environment to make the resulting timed automaton easy to visualize. Let us consider the timed automaton from the abstraction in Fig. 5.1 and the LTL formula given by the following,

$$\phi_1 = (\neg A \mathbf{U} B) \wedge (\diamond A).$$

This specification requires the robot to visit the area B first and eventually visit A also. The resulting automaton based on the method in the previous section is as shown in Fig. 5.6. Each state corresponds to a product state between a state in \mathcal{T}_{map} and a state in TT \mathcal{A} . The Büchi accepting states are indicated by an additional b in their state names. We obtained the optimal path by first adding a final state and linking every accepting state to it, and then using UPPAAL to find one of the shortest paths that satisfies the condition “ $E \llcorner final$ ”. UPPAAL will then

¹The tool is available on <https://github.com/yzh89/MITL2Timed>

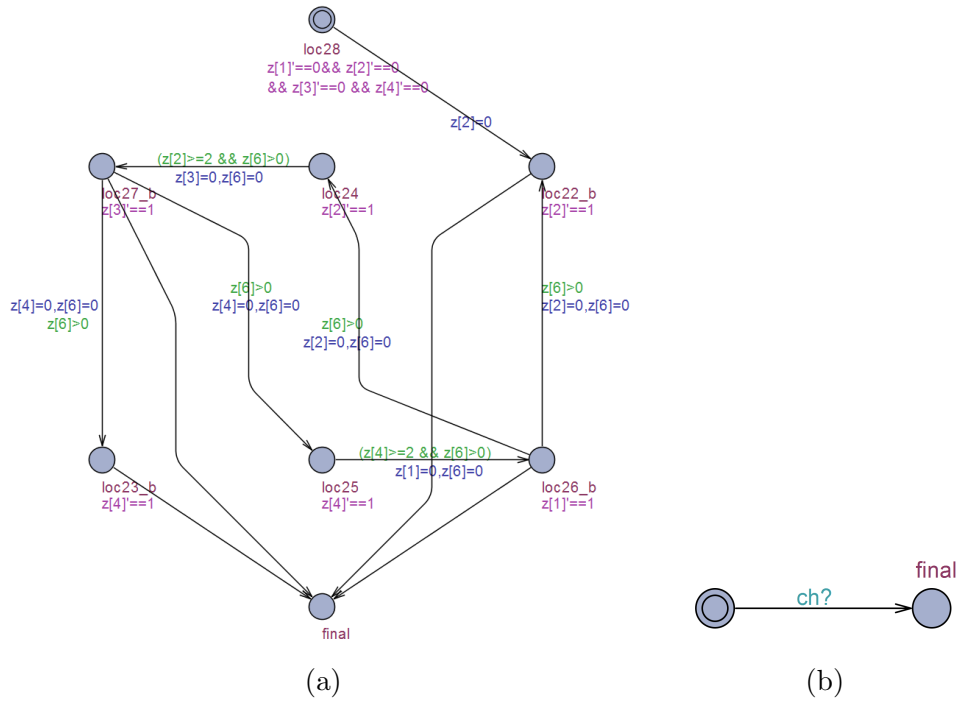


Figure 5.7: Fig. (a) shows the other timed automata of ϕ_2 corresponding to the generator of timed eventually. Fig. (b) shows the verification timed automaton, that checks if the falling edge of generator is ever detected, i.e. if a synchronization signal ($ch!$) has happened. This signal marks the end of a full eventually cycle. Similar to the LTL case, we ask UPPAAL to check for us the following property, if there is a trajectory that leads to the final states in (a) and (b). The optimal path in this case is $(loc19, loc28) \rightarrow (loc3, loc28) \rightarrow (loc3, loc22_b)$. The states are products of states of Fig. 5.8 and Fig. (a). This path corresponds to $(pos0, t \in [0, 1]) \rightarrow (pos3 : A, t \in [1, 2]) \rightarrow (pos0, t \in [2, 3])$ in physical space. Repeating this path will satisfy ϕ_2 .

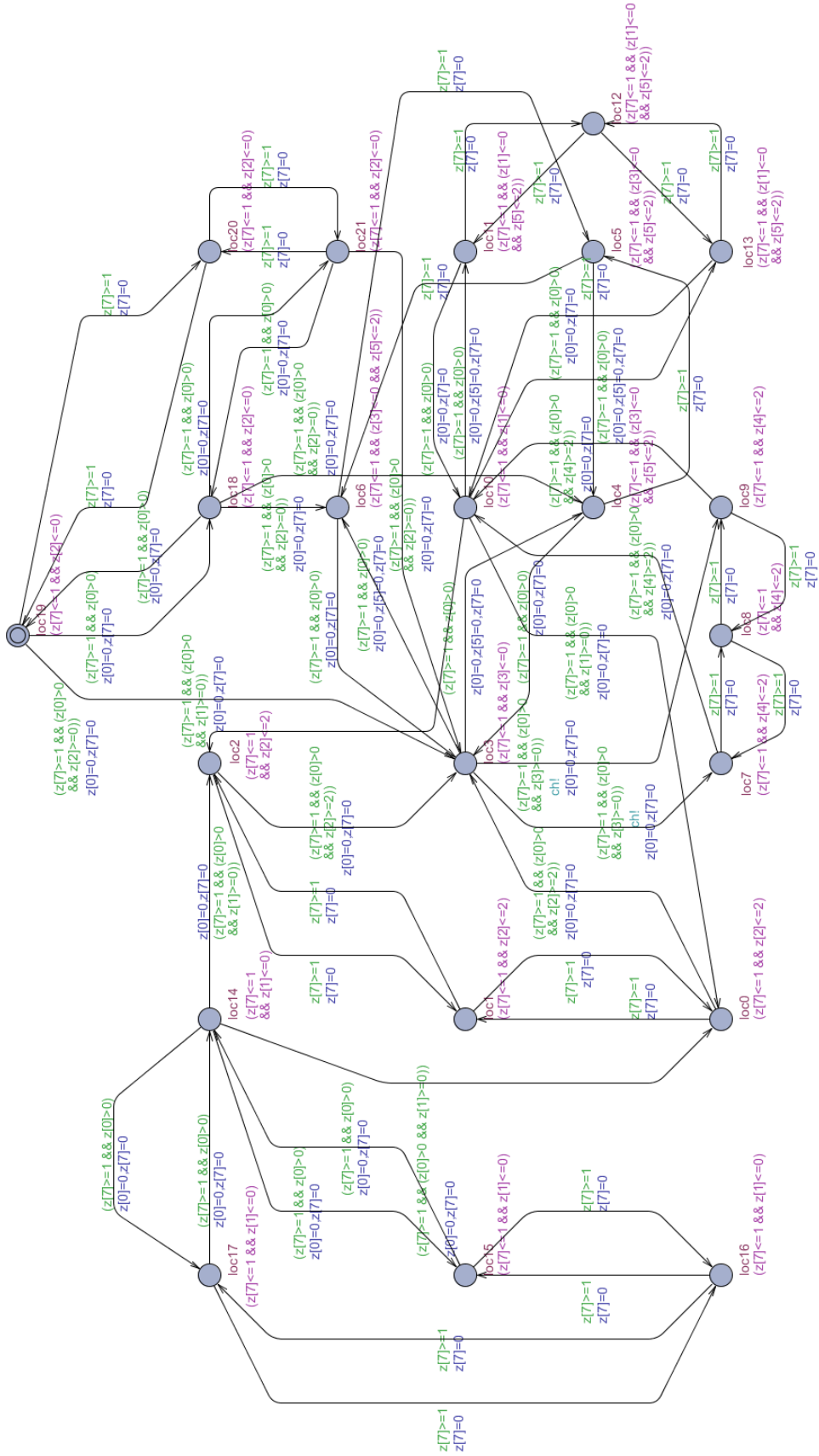


Figure 5.8: This shows one of the resulting timed automata in UPPAAL of ϕ_2 corresponding to the checker of timed eventually operator and untimed always. Some of the edges are further annotated by synchronization signal (ch!).

compute one fastest path in the timed automaton that goes to final state, provided such a path exists. If such a path exists, then it is a finite trajectory that satisfies the specifications. In this work, we are more interested in planning a path that satisfies some MITL constraint, so a finite time trajectory is a valid solution. The initial states of the automaton is *loc0* which is the only state at *pos0* that outputs 1. The optimal trajectory is $loc0 \rightarrow loc2 \rightarrow loc7 \rightarrow loc6_b$, in the product automaton. This means that the optimal way for a robot to satisfy the LTL constraint is to traverse the map in the following order, $pos0 \rightarrow pos1 : B \rightarrow pos0 \rightarrow pos3 : A$.

In the second test case, the environment stays the same and the requirement is captured in a MITL formula ϕ_2

$$\phi_2 = \square \diamond_{[0,2]} A$$

This requires the robot to perform periodic survey of area *A* every 2s. The resulting timed automata are shown in Fig. 5.8 and Fig. 5.7. As we discussed earlier, if a synchronization signal (*ch!*) is sent, the falling edge for the output of the generator automaton is detected and verified. This marks the end of a finite trajectory that satisfies the MITL constraints. We used the automaton in Fig. 5.7 (b) to receive such signal. Similar to the LTL case, we use UPPAAL to find one fastest path that leads to the final state in Fig. 5.7(a) and 5.7(b).

The optimal trajectory in this case is $(loc19, loc28) \rightarrow (loc3, loc28) \rightarrow (loc3, loc22_b)$, which corresponds to $(pos0, t \in [0, 1]) \rightarrow (pos3 : A, t \in [1, 2]) \rightarrow (pos0, t \in [2, 3])$ in physical space. Repeating this path will satisfy ϕ_2 .

All the computations are done on a computer with 3.4GHz processor and 8GB

Table 5.1: Computation Time for typical MITL formula

MITL Formula	Map Grid	Transformation Time	Number of TA Transitions	Synthesis Time
ϕ_1	2x2	$< 0.001s$	22	0.016s
ϕ_2	2x2	$0.004s$	69	0.018s
ϕ_3	2x2	$0.40s$	532	0.10s
ϕ_4	2x2	$0.46s$	681	0.12s
ϕ_1	4x4	$0.004s$	181	0.062s
ϕ_1	8x8	$0.015s$	886	0.21s
ϕ_2	8x8	$0.015s$	1795	0.32s

memory. Both of the previous examples require very small amount of time ($< 0.03s$).

We also tested our implementation against various other complex environments and MITL formulas. Table 5.1 summarizes our results for complex systems and formulas. The map we demonstrated earlier is a 2x2 map (Fig. 5.1); we also examined the cases for 4x4 and 8x8 grid maps. The other temporal logic formulas used are listed below. The time intervals in the formula are scaled according to the map size.

$$\phi_3 = \diamond_{[0,4]}A \wedge \diamond_{[0,4]}B$$

$$\phi_4 = \diamond_{[2,4]}A \wedge \diamond_{[0,2]}B$$

It can be seen from Table 5.1 that our algorithm performs very well with common MITL formulas and scales satisfactorily with the dimensions of the map.

5.5 Continuous Trajectory generation

In this section, we briefly describe the generation of a continuous trajectory from the discrete motion plan obtained from the timed automaton. Let us consider

Since our environment is decomposed in rectangular cells, the robot only needs to move forward, turn right, turn left and make a U-turn. We synthesize a controller that can make the robot to perform these elementary motion segments within the given time.

For moving forward, the input ω is chosen to be 0 and the velocity u is tuned so that the robot reaches the final position in time. For turning left and turning right, ω is chosen to take positive and negative values respectively so that a circular arc is traversed. Similarly the U-turn is also implemented so that the robot performs the U-turn within a single cell.

Let us denote the state of the system at time t by the pair (q, t) i.e. $x(t) = q_1$, $y(t) = q_2$ and $\theta(t) = q_3$ where $q = [q_1, q_2, q_3]$. Then we have the following lemma on the optimality of the control inputs.

Lemma 5.5.1. *If $\bar{u}(t)$ and $\bar{\omega}(t)$, $t \in [0, 1]$ is a pair of control inputs s.t. the dynamics moves from state $(q_0, 0)$ to $(q_1, 1)$, then $u(t_0 + t) = \frac{1}{\lambda}\bar{u}(\frac{t}{\lambda})$ and $\omega(t_0 + t) = \frac{1}{\lambda}\bar{\omega}(\frac{t}{\lambda})$ moves the system from (q_0, t_0) to $(q_1, t_0 + \lambda)$ for any $\lambda > 0$.*

Moreover, if \bar{u} and $\bar{\omega}$ move the system optimally, i.e.

$$J(\bar{u}, \bar{\omega}) = \min_{u(\cdot), w(\cdot)} \int_0^1 [r_1 u^2(t) + r_2 w^2(t)] dt \quad (5.2)$$

then u and ω given above are also optimal for moving the system from (q_0, t_0) to $(q_1, t_0 + \lambda)$, i.e.

$$J_1(u, \omega) = \min_{u_1(\cdot), w_1(\cdot)} \int_{t_0}^{t_0 + \lambda} [r_1 u_1^2(t) + r_2 w_1^2(t)] dt. \quad (5.3)$$

Proof. Let us first denote

$$G(q) = \begin{bmatrix} \cos(\theta(t)) & 0 \\ \sin(\theta(t)) & 0 \\ 0 & 1 \end{bmatrix}$$

where $q = [x(t), y(t), \theta(t)]$. Therefore, dynamics (5.1) can be written as $\dot{q} =$

$G(q) \begin{bmatrix} u \\ \omega \end{bmatrix}$. Let us now consider $\bar{q}(t) = [x(t_0 + \lambda t), y(t_0 + \lambda t), \theta(t_0 + \lambda t)]$. Therefore,

$\dot{\bar{q}} = \lambda G(\bar{q}) \begin{bmatrix} u(t_0 + \lambda t) \\ \omega(t_0 + \lambda t) \end{bmatrix}$. Using the definition of u and ω in the lemma, we get $\dot{\bar{q}} =$

$G(\bar{q}) \begin{bmatrix} \bar{u} \\ \bar{\omega} \end{bmatrix}$. By the hypothesis of the lemma, \bar{u} and $\bar{\omega}$ move the system from $(q_0, 0)$ to $(q_1, 1)$ i.e. from $[x(t_0), y(t_0), \theta(t_0)] = q_0$ to $q_1 = \bar{q}(1) = [x(t_0 + \lambda), y(t_0 + \lambda), \theta(t_0 + \lambda)]$.

For optimality, let the proposed u, ω be not optimal and u^* and ω^* are optimal ones i.e.

$$\int_{t_0}^{t_0+\lambda} [r_1 u^{*2}(t) + r_2 \omega^{*2}(t)] dt \leq \int_{t_0}^{t_0+\lambda} [r_1 u^2(t) + r_2 \omega^2(t)] dt \quad (5.4)$$

Now let us construct $\bar{u}^*(t) = \lambda u^*(t_0 + \lambda t)$ and $\bar{\omega}^*(t) = \lambda \omega^*(t_0 + \lambda t)$.

Therefore from (5.4),

$$\int_0^1 [r_1 \bar{u}^{*2}(t_0 + \lambda s) + r_2 \bar{\omega}^{*2}(t_0 + \lambda s)] ds \leq \int_0^1 [r_1 u^2(t_0 + \lambda s) + r_2 \omega^2(t_0 + \lambda s)] ds$$

$$\int_0^1 [r_1 \bar{u}^{*2}(s) + r_2 \bar{\omega}^{*2}(s)] ds \leq \int_0^1 [r_1 \bar{u}^2(s) + r_2 \bar{\omega}^2(s)] ds \quad (5.5)$$

But, by the hypothesis, \bar{u} and $\bar{\omega}$ are optimal and hence

$$\int_0^1 [r_1 \bar{u}^{*2}(s) + r_2 \bar{\omega}^{*2}(s)] ds \geq \int_0^1 [r_1 \bar{u}^2(s) + r_2 \bar{\omega}^2(s)] ds \quad (5.6)$$

Combining (5.5) and (5.6) we get,

$$\int_0^1 [r_1 \bar{u}^{*2}(s) + r_2 \bar{\omega}^{*2}(s)] ds = \int_0^1 [r_1 \bar{u}^2(s) + r_2 \bar{\omega}^2(s)] ds \quad (5.7)$$

After changing the dummy variables inside integration again, one can obtain

$$\int_{t_0}^{t_0+\lambda} [r_1 u^{*2}(s) + r_2 \omega^{*2}(s)] ds = \int_{t_0}^{t_0+\lambda} [r_1 u^2(s) + r_2 \omega^2(s)] ds \quad (5.8)$$

Hence the proposed u and ω are optimal whenever \bar{u} and $\bar{\omega}$ are optimal. \square

Remark 5.5.2. *Lemma 5.5.1 states that if the controls for elementary motion from initial time 0 to final time 1 are synthesized, then by properly scaling and shifting in the time and scaling the magnitude, controls for any movement from any initial time to any final time can be synthesized without further solving any optimization problem.*

5.6 Conclusion

In this work, we have presented a timed automaton based approach to generate a discrete plan for the robot to perform temporal tasks with finite time constraints. We implemented the algorithm in an efficient and generic way so that it can translate the time constraints and temporal specifications to timed automaton models in UPPAAL and synthesize the path accordingly. We then demonstrated our algorithm in grid type environments with different MITL formulas. We have considered grid

type of environment for our case studies, but the approach can be generalized to most motion planning problems as long as the environment can be decomposed into cells. We also provide a brief overview on generating an optimal continuous trajectory from the discrete plan. For future works, we are considering to extend the work to include dynamic obstacles as well as multiagent systems.

Chapter 6: Conclusion

To conclude, our research focuses on two aspects of motion planning and controls, One is to answer the question of how to verify the safety of the trajectory planner and controls, as well as how to synthesize an safe planner and controller. The other is to answer how to generate a high level plan to satisfy temporal specifications.

In the first aspect of our work, we proposed a verification method to prove the safety and robustness of a path planner and the path following controls based on reachable sets. Secondly, we proposed a reachable set based collision avoidance algorithm for UAVs. Instead of the traditional approaches of collision avoidance between trajectories, we proposed a collision avoidance scheme based on reachable sets and tubes. We formulated the problem as a convex optimization problem seeking control set design for the aircraft to avoid collision. We applied our approach to collision avoidance scenarios of quadrotors and fixed-wing aircraft.

In the second aspect of our work, we addressed the high level planning problems with timed temporal logic constraints. Firstly, we presented an optimization based method for path planning of a mobile robot subject to timed temporal constraints, in a dynamic environment. Temporal logic (TL) can address very complex

task specifications such as safety, coverage, motion sequencing etc. We used metric temporal logic (MTL) to encode the task specifications with timing constraints. We translated the MTL formulae into mixed integer linear constraints and solved the associated optimization problem using a mixed integer linear program solver. We applied our approach on several case studies in complex dynamical environments subjected to timed temporal specifications. Secondly, we presented a timed automaton based method for planning under the given timed temporal logic specifications. We used metric interval temporal logic (MITL), a member of the MTL family, to represent the task specification, and provided a constructive way to generate a timed automaton and methods to look for accepting runs on the automaton to find an optimal motion (or path) sequence for the robot to complete the task.

Appendix A:

Publications

- [1] Shahan Yang, Yuchen Zhou, and John Baras. Compositional analysis of dynamic bayesian networks and applications to complex dynamic system decomposition. *Procedia Computer Science*, 16:167–176, 2013.
- [2] Yuchen Zhou and John S Baras. CPS modeling integration hub and design space exploration with application to microrobotics. In *Control of Cyber-Physical Systems*, pages 23–42. Springer International Publishing, 2013.
- [3] Yuchen Zhou, John Baras, and Shige Wang. Hardware software co-design for automotive cps using architecture analysis and design language. In *the 5th Analytic Virtual Integration of Cyber-Physical Systems Workshop*, 2014.
- [4] Jacob Moschler, Yuchen Zhou, John S. Baras, and Jungwoo Joh. A systems engineering approach to collaborative coordination of UAS in the NAS with safety guarantees. In *Integrated Communications, Navigation and Surveillance Conference (ICNS), 2014*, pages U1–1. IEEE, 2014.
- [5] Yuchen Zhou, Dipankar Maity, and John S. Baras. Optimal Mission Planner with Timed Temporal Logic Constraints. In *European Control Conference 2015 (ECC'15)*. IEEE, 2015.
- [6] Yuchen Zhou and John S. Baras. Reachable set approach to collision avoidance for UAVs. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 5947–5952, Dec 2015.
- [7] Yuchen Zhou, Dipankar Maity, and John S Baras. Timed automata approach for motion planning using metric interval temporal logic. In *European Control Conference 2016 (ECC'16)*, 2016.

Bibliography

- [1] Steven Michael LaValle. *Planning algorithms*. Cambridge university press, 2006.
- [2] B. Fleming. New Automotive Electronics Technologies [Automotive Electronics]. *IEEE Vehicular Technology Magazine*, 7(4):4–12, December 2012.
- [3] ‘Super Cruise’ Takes on Real-World Traffic Scenarios. <https://media.gm.com/media/us/en/gm/news.detail.html/content/Pages/news/us/en/2013/Apr/0429-cadillac-super-cruise.html>, April 2013.
- [4] Aaron M. Kessler. Elon Musk Says Self-Driving Tesla Cars Will Be in the US by Summer. *The New York Times*. <http://mobile.nytimes.com/2015/03/20/business/elon-musk-says-selfdriving-tesla-cars-will-be-in-the-us-by-summer.html>, 2015.
- [5] Bruce Donald, Patrick Xavier, John Canny, and John Reif. Kinodynamic motion planning. *Journal of the ACM (JACM)*, 40(5):1048–1066, 1993.
- [6] Takashi Maekawa, Tetsuya Noda, Shigefumi Tamura, Tomonori Ozaki, and Ken-ichiro Machida. Curvature continuous path generation for autonomous vehicle using B-spline curves. *Computer-Aided Design*, 42(4):350–359, 2010.
- [7] David J. Webb and Jan van den Berg. Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics. In *2013 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5054–5061. IEEE, 2013.
- [8] Naomi Ehrich Leonard and Edward Fiorelli. Virtual leaders, artificial potentials and coordinated control of groups. In *Proceedings of the 40th IEEE Conference on Decision and Control, 2001.*, volume 3, pages 2968–2973. IEEE, 2001.
- [9] Yoram Koren and Johann Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *Proceedings of 1991 IEEE International Conference on Robotics and Automation, 1991.*, pages 1398–1404. IEEE, 1991.

- [10] Karin Sigurd and Jonathan How. UAV Trajectory Design Using Total Field Collision Avoidance. American Institute of Aeronautics and Astronautics, August 2003.
- [11] C. Carbone, U. Ciniglio, F. Corraro, and S. Luongo. A novel 3d geometric algorithm for aircraft autonomous collision avoidance. In *2006 45th IEEE Conference on Decision and Control*,, pages 1580–1585. IEEE, 2006.
- [12] Emmett Lalish, Kristi A. Morgansen, and Takashi Tsukamaki. Decentralized reactive collision avoidance for multiple unicycle-type vehicles. In *American Control Conference, 2008*, pages 5055–5061. IEEE, 2008.
- [13] Su-Cheol Han and Hyochoong Bang. Proportional navigation-based optimal collision avoidance for uavs. In *2nd International Conference on Autonomous Robots and Agents*, pages 13–15, 2004.
- [14] I.M. Mitchell, A.M. Bayen, and C.J. 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.
- [15] Michael P. Vitus and Claire J. Tomlin. Hierarchical, Hybrid Framework for Collision Avoidance Algorithms in the National Airspace. *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2008.
- [16] Alexander B. Kurzhanski and Pravin Varaiya. Ellipsoidal Techniques for Reachability Analysis. In Nancy Lynch and Bruce H. Krogh, editors, *Hybrid Systems: Computation and Control*, number 1790 in Lecture Notes in Computer Science, pages 202–214. Springer Berlin Heidelberg, 2000.
- [17] Oleg Botchkarev and Stavros Tripakis. Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations. In *Hybrid Systems: Computation and Control*, pages 73–88. Springer, 2000.
- [18] Colas Le Guernic and Antoine Girard. Reachability analysis of linear systems using support functions. *Nonlinear Analysis: Hybrid Systems*, 4(2):250–262, 2010.
- [19] Matthias Althoff, Olaf Stursberg, and Martin Buss. Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization. In *47th IEEE Conference on Decision and Control CDC 2008.*, pages 4042–4048, 2008.
- [20] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. SpaceEx: Scalable verification of hybrid systems. In *Computer Aided Verification*, pages 379–395, 2011.

- [21] Antoine Girard, Colas Le Guernic, and Oded Maler. Efficient computation of reachable sets of linear time-invariant systems with inputs. In *Hybrid Systems: Computation and Control*, pages 257–271. Springer, 2006.
- [22] Eugene Asarin, Olivier Bournez, Thao Dang, and Oded Maler. Approximate reachability analysis of piecewise-linear dynamical systems. In *Hybrid Systems: Computation and Control*, pages 20–31. Springer, 2000.
- [23] Matthias Althoff. Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets. In *Proceedings of the 16th international conference on Hybrid systems: computation and control*, pages 173–182. ACM, 2013.
- [24] Eugene Asarin, Thao Dang, Goran Frehse, Antoine Girard, Colas Le Guernic, and Oded Maler. Recent progress in continuous and hybrid reachability analysis. In *2006 IEEE Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control*, pages 1582–1587. IEEE, 2006.
- [25] Houssam Abbas, Georgios Fainekos, Sriram Sankaranarayanan, Franjo Ivan-textbackslashvčić, and Aarti Gupta. Probabilistic temporal logic falsification of cyber-physical systems. *ACM Transactions on Embedded Computing Systems (TECS)*, 12(2s):95, 2013.
- [26] Luis I. Reyes Castro, Pratik Chaudhari, Jana Tumova, Sertac Karaman, Emilio Frazzoli, and Daniela Rus. Incremental Sampling-based Algorithm for Minimum-violation Motion Planning. *arXiv preprint arXiv:1305.1102*, 2013.
- [27] Yushan Chen, Xu Chu Ding, Alin Stefanescu, and Calin Belta. A formal approach to deployment of robotic teams in an urban-like environment. In *Distributed Autonomous Robotic Systems*, pages 313–327. Springer, 2013.
- [28] Meng Guo and Dimos V. Dimarogonas. Reconfiguration in Motion Planning of Single- and Multi-agent Systems under Infeasible Local LTL Specifications. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Tokyo Big Sight, Japan, 2013.
- [29] Kangjin Kim and Georgios Fainekos. Minimal Specification Revision for Weighted Transition Systems. In *2013 IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, 2013.
- [30] Matthew R. Maly, Morteza Lahijanian, Lydia E. Kavraki, Hadas Kress-Gazit, and Moshe Y. Vardi. Iterative temporal motion planning for hybrid systems in partially unknown environments. In *Proceedings of the 16th international conference on Hybrid systems: computation and control*, pages 353–362, 2013.
- [31] Jorge A. Baier, Christian Fritz, Meghyn Bienvenu, and Sheila A. McIlraith. Beyond classical planning: Procedural control knowledge and preferences in

- state-of-the-art planners revisited. In *Proceedings of the ICAPS 2009 Workshop on Generalized Planning: Macros, Loops, Domain Control.*, Thessaloniki, Greece, 2009.
- [32] Edmund M Clarke, Orna Grumberg, and Doron Peled. *Model checking*. MIT press, 1999.
- [33] M.M. Quottrup, T. Bak, and R.I. Zamanabadi. Multi-robot planning : a timed automata approach. In *2004 IEEE International Conference on Robotics and Automation (ICRA)*, volume 5, pages 4417–4422 Vol.5, 2004.
- [34] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, Cambridge, Mass., 2008.
- [35] Gerard J. Holzmann. The model checker SPIN. *IEEE Transactions on software engineering*, (5):279–295, 1997.
- [36] Alessandro Cimatti, Edmund Clarke, Enrico Giunchiglia, Fausto Giunchiglia, Marco Pistore, Marco Roveri, Roberto Sebastiani, and Armando Tacchella. Nusmv 2: An opensource tool for symbolic model checking. In *Computer Aided Verification*, pages 359–364. Springer, 2002.
- [37] Kamal Kant and Steven W Zucker. Toward efficient trajectory planning: The path-velocity decomposition. *The International Journal of Robotics Research*, 5(3):72–89, 1986.
- [38] Michael Erdmann and Tomas Lozano-Perez. On multiple moving objects. *Algorithmica*, 2(1-4):477–521, 1987.
- [39] H. Jin Kim, David H. Shim, and Shankar Sastry. Nonlinear model predictive tracking control for rotorcraft-based unmanned aerial vehicles. In *Proceedings of the 2002 American Control Conference, 2002.*, volume 5, pages 3576–3581. IEEE, 2002.
- [40] D. Limon, I. Alvarado, T. Alamo, and E. F. Camacho. Robust tube-based MPC for tracking of constrained linear systems with additive disturbances. *Journal of Process Control*, 20(3):248–260, 2010.
- [41] DongBin Lee, C. Nataraj, Timothy C. Burg, and Darren M. Dawson. Adaptive tracking control of an underactuated aerial vehicle. In *American Control Conference (ACC), 2011*, pages 2326–2331. IEEE, 2011.
- [42] Jacob Moschler, Yuchen Zhou, John S. Baras, and Jungwoo Joh. A systems engineering approach to collaborative coordination of UAS in the NAS with safety guarantees. In *Integrated Communications, Navigation and Surveillance Conference (ICNS), 2014*, pages U1–1. IEEE, 2014.
- [43] Colas Le Guernic. *Reachability Analysis of Hybrid Systems with Linear Continuous Dynamics*. PhD thesis, Université Joseph Fourier, 2009.

- [44] Alongkritt Chutinan and Bruce H. Krogh. Computational techniques for hybrid system verification. *IEEE Transactions on Automatic Control*, 48(1):64–75, 2003.
- [45] Matthias Althoff, Daniel Althoff, Dirk Wollherr, and Martin Buss. Safety verification of autonomous vehicles for coordinated evasive maneuvers. In *Intelligent vehicles symposium (IV), 2010 IEEE*, pages 1078–1083. IEEE, 2010.
- [46] M. Althoff and J.M. Dolan. Set-based computation of vehicle behaviors for the online verification of autonomous vehicles. In *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1162–1167, 2011.
- [47] Matthias Althoff and John M. Dolan. Reachability computation of low-order models for the safety verification of high-order road vehicle models. In *American Control Conference (ACC), 2012*, pages 3559–3566. IEEE, 2012.
- [48] Matthias Althoff. *Reachability analysis and its application to the safety assessment of autonomous cars*. PhD thesis, 2010.
- [49] Antoine Girard. Reachability of uncertain linear systems using zonotopes. In *Hybrid systems: computation and control*, pages 291–305. Springer, 2005.
- [50] Rajeev Alur, Costas Courcoubetis, Thomas A. Henzinger, and Pei-Hsin Ho. *Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems*. Springer, 1993.
- [51] Thao Dang, Colas Le Guernic, and Oded Maler. Computing reachable states for nonlinear biological models. *Theoretical Computer Science*, 412(21):2095–2107, 2011.
- [52] C.J. Tomlin, I. Mitchell, A.M. Bayen, and M. Oishi. Computational techniques for the verification of hybrid systems. *Proceedings of the IEEE*, 91(7):986–1001, July 2003.
- [53] Meeko Oishi, Ian Mitchell, Claire Tomlin, and Patrick Saint-Pierre. Computing viable sets and reachable sets to design feedback linearizing control laws under saturation. In *2006 45th IEEE Conference on Decision and Control*, pages 3801–3807. IEEE, 2006.
- [54] Gabriel M. Hoffmann, Steven L. Waslander, and Claire J. Tomlin. Quadrotor helicopter trajectory tracking control. In *AIAA Guidance, Navigation and Control Conference and Exhibit, Honolulu, Hawaii*, pages 1–14, 2008.
- [55] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2520–2525. IEEE, 2011.

- [56] Daniel Mellinger, Nathan Michael, and Vijay Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors. *The International Journal of Robotics Research*, 31(5):664–674, 2012.
- [57] Jin-woo Lee. *Method of path planning for evasive steering maneuver*. Google Patents, January 2016.
- [58] Jin-Woo Lee and Bakhtiar Litkouhi. A unified framework of the automated lane centering/changing control for motion smoothness adaptation. In *2012 15th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 282–287. IEEE, 2012.
- [59] Jesse Levinson, Jake Askeland, Jan Becker, Jennifer Dolson, David Held, Soeren Kammel, J. Zico Kolter, Dirk Langer, Oliver Pink, Vaughan Pratt, and others. Towards fully autonomous driving: Systems and algorithms. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 163–168. IEEE, 2011.
- [60] Wei Xi and John S. Baras. MPC based motion control of car-like vehicle swarms. In *Mediterranean Conference on Control & Automation MED’07.*, pages 1–6. IEEE, 2007.
- [61] Nikolai K. Moshchuk, Shih-ken Chen, and Chad T. Zagorski. *COLLISION AVOIDANCE CONTROL INTEGRATED WITH EPS CONTROLLER*. June 2015.
- [62] Yuchen Zhou and John S. Baras. Reachable set approach to collision avoidance for UAVs. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 5947–5952, Dec 2015.
- [63] A. A. Kurzhanskiy and P. Varaiya. Ellipsoidal Toolbox. Technical Report UCB/EECS-2006-46, EECS Department, University of California, Berkeley, May 2006.
- [64] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.
- [65] Michael Grant and Stephen Boyd. CVX: Matlab Software for Disciplined Convex Programming, version 2.1. March 2014.
- [66] Nathan Michael, Daniel Mellinger, Quentin Lindsey, and Vijay Kumar. The GRASP Multiple Micro-UAV Testbed. *IEEE Robotics & Automation Magazine*, 17(3):56–65, September 2010.
- [67] Luis Rodolfo Garcia Carrillo, Alejandro Enrique Dzul López, Rogelio Lozano, and Claude Pégard. Modeling the Quadrotor Mini-Rotorcraft. *Quad Rotorcraft Control*, pages 23–34, 2013.

- [68] Eric M Wolff, Ufuk Topcu, and Richard M Murray. Optimization-based Trajectory Generation with Linear Temporal Logic Specifications. In *Int. Conf. on Robotics and Automation*, 2014.
- [69] Randal W Beard and Timothy W McLain. *Small unmanned aircraft: Theory and practice*. Princeton University Press, 2012.
- [70] Michael Grant and Stephen Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pages 95–110. Springer-Verlag Limited, 2008.
- [71] J.C. Latombe. *Robot motion planning*. Kluwer international series in engineering and computer science;124. Kluwer Academic Publishers, Boston, cop. 1991.
- [72] Howie Choset. Coverage for robotics—A survey of recent results. *Annals of mathematics and artificial intelligence*, 31(1-4):113–126, 2001.
- [73] Howie M Choset. *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005.
- [74] Wei Xi, Xiaobo Tan, and John S Baras. A hybrid scheme for distributed control of autonomous swarms. In *Proceedings of the 2005 American Control Conference.*, pages 3486–3491. IEEE, 2005.
- [75] Rajeev Sharma. Locally efficient path planning in an uncertain, dynamic environment using a probabilistic model. *Robotics and Automation, IEEE Transactions on*, 8(1):105–110, 1992.
- [76] Steven M LaValle and Rajeev Sharma. On motion planning in changing, partially predictable environments. *The International Journal of Robotics Research*, 16(6):775–805, 1997.
- [77] H. Kress-Gazit, G.E. Fainekos, and G.J. Pappas. Temporal-Logic-Based Reactive Mission and Motion Planning. *IEEE Transactions on Robotics*, 25(6):1370–1381, December 2009.
- [78] Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-time systems*, 2(4):255–299, 1990.
- [79] Joël Ouaknine and James Worrell. Some recent results in metric temporal logic. In *Formal Modeling and Analysis of Timed Systems*, pages 1–13. Springer, 2008.
- [80] Sertac Karaman, Ricardo G Sanfelice, and Emilio Frazzoli. Optimal control of mixed logical dynamical systems with linear temporal logic specifications. In *47th IEEE Conference on Decision and Control*, pages 2117–2122. IEEE, 2008.

- [81] Sertac Karaman and Emilio Frazzoli. Vehicle routing problem with metric temporal logic specifications. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 3953–3958. IEEE, 2008.
- [82] Arthur Richards and Jonathan P. How. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *Proceedings of the 2002 American Control Conference*, volume 3, pages 1936–1941. IEEE, 2002.
- [83] Yuchen Zhou, Dipankar Maity, and John S. Baras. Optimal Mission Planner with Timed Temporal Logic Constraints. In *European Control Conference 2015 (ECC'15)*. IEEE, 2015.
- [84] D. Maity and J.S. Baras. Motion planning in dynamic environments with bounded time temporal logic specifications. In *2015 23th Mediterranean Conference on Control and Automation (MED)*, pages 940–946, June 2015.
- [85] Rajeev Alur, Tomás Feder, and Thomas A. Henzinger. The benefits of relaxing punctuality. *Journal of the ACM (JACM)*, 43(1):116–146, 1996.
- [86] Joël Ouaknine and James Worrell. On the decidability of metric temporal logic. In *Proceedings. 20th Annual IEEE Symposium on Logic in Computer Science 2005*, pages 188–197. IEEE, 2005.
- [87] Mark Jenkins, Joël Ouaknine, Alexander Rabinovich, and James Worrell. Alternating timed automata over bounded time. In *2010 25th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 60–69. IEEE, 2010.
- [88] Dejan Ničković and Nir Piterman. *From MTL to deterministic timed automata*. Springer, 2010.
- [89] Oded Maler, Dejan Nickovic, and Amir Pnueli. From MITL to Timed Automata. In Eugene Asarin and Patricia Bouyer, editors, *Formal Modeling and Analysis of Timed Systems*, number 4202 in Lecture Notes in Computer Science, pages 274–289. Springer Berlin Heidelberg, 2006.
- [90] Glenn Behrmann, Alexandre David, Kim G Larsen, John Hakansson, Paul Pettersson, Wang Yi, and Monique Hendriks. UPPAAL 4.0. In *Third International Conference on Quantitative Evaluation of Systems, 2006. QEST 2006.*, pages 125–126. IEEE, 2006.
- [91] Yuchen Zhou, Dipankar Maity, and John S Baras. Timed automata approach for motion planning using metric interval temporal logic. In *European Control Conference 2016 (ECC'16)*, 2016.
- [92] Stephen L. Smith, Jana Tumova, Calin Belta, and Daniela Rus. Optimal path planning under temporal logic constraints. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3288–3293, 2010.

- [93] Eric M. Wolff, Ufuk Topcu, and Richard M. Murray. Automaton-Guided Controller Synthesis for Nonlinear Systems with Temporal Logic. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Tokyo Big Sight, Japan, 2013.
- [94] A. Prasad Sistla and Edmund M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM (JACM)*, 32(3):733–749, 1985.
- [95] Patricia Bouyer, Nicolas Markey, Joël Ouaknine, and James Worrell. On expressiveness and complexity in real-time model checking. In *Automata, Languages and Programming*, pages 124–135. Springer, 2008.
- [96] Paul Gastin and Denis Oddoux. Fast LTL to Büchi Automata Translation. In Gérard Berry, Hubert Comon, and Alain Finkel, editors, *Computer Aided Verification*, number 2102 in Lecture Notes in Computer Science, pages 53–65. Springer Berlin Heidelberg, 2001.