UNIVERSITY OF CALIFORNIA
SANTA CRUZ

MOTION PLANNING TO AVOID OBSTACLES WITH HYBRID DYNAMICS

A thesis submitted in partial satisfaction of the
requirements for the degree of

Master of Science

in

COMPUTER ENGINEERING
with an emphasis in ROBOTICS AND CONTROL

by

Adam Ames

June 2021

The Thesis of Adam Ames
is approved:

Professor Ricardo G. Sanfelice, Chair

Professor Gabriel Elkaim

Professor Abhishek Halder

Quentin Williams
Interim Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

Abstract

Motion Planning to Avoid Obstacles with Hybrid Dynamics

by

Adam Ames

This paper proposes a set-based feedback motion planning algorithm to drive a quadrotor to a target set while avoiding dynamic obstacles which behave like a bouncing ball. The planner makes use of a library of motion primitives to allow for complex maneuvers and fast planning times. The planner is capable of avoiding obstacles with unknown angular velocities which change the obstacle velocity at bounces, resulting in safe trajectories with the use of safety margins. Simulation and experimental data of the planner are presented.

Chapter 1

Introduction

Quadrotors have been the focus of increasing amounts of research with their popularity as an autonomous vehicle platform continuing to grow. A large portion of this research has been focused on solving the motion planning and obstacle avoidance problem. There are been many strategies ranging from velocity fields, like that proposed in [10], to graph search algorithms like in [3]. One popular approach is receding horizon control [1] [7], where the motion planning problem is converted to a constrained optimization problem. Obstacle avoidance is achieved through hard constraints on the optimization with soft constraints selecting from the safe trajectories. While the collision avoidance problem has been well studied, the case when obstacles bounce has been largely neglected. To the best of the author's knowledge there is only one currently published paper which includes the bouncing obstacle case. In [6], a Nonlinear MPC approach is used to generate a series of safe control inputs for a quadrotor with obstacles allowed to move in a line or behave like a bouncing ball. Our approach differs in the obstacle model and the planning methodology. The proposed planner generates a reference trajectory instead of a control input, and the bouncing ball model used also includes the influence of the obstacle's angular velocity at impact.

1.1 Preliminaries on Hybrid Systems

A hybrid dynamical system is a system with both continuous and discrete dynamics. The hybrid system $\mathcal{H}$ for the state $x \in R^n$ and input $u \in \mathbb{R}^m$ can be modeled as, following [5],

$$\mathcal{H} := \begin{cases} \dot{x} \quad \in F(x,u) & (x,u) \in C \\ x^+ \in G(x,u) & (x,u) \in D \end{cases} \tag{1.1}$$

The data of the system $\mathcal{H} = (C, D, F, G)$ describe how the system evolves over time. The flow map F describes the continuous evolution of x while the state and input are in the flow set C. The jump map G describes the discrete evolution of x while the state and input are in the jump set D.

A hybrid time domain $E \subset \mathbb{R}_{\geq 0} \times \mathbb{N}$ is defined as union of intervals $E \cap ([0,T] \times \{0,1,\ldots,J\}) = \bigcup_{j=0}^{J-1}([t_j, t_{j+1}], j)$ for each $(T,J) \in E$, with the time interval $t_0 = 0 \leq t_1 \leq t_2 \leq \ldots \leq t_J = T$. A hybrid arc $\phi : \mathbb{R}_{\geq 0} \times \mathbb{N} \to \mathbb{R}^n$ is parameterized by regular time $t \in [0,T]$ and discrete number of jumps $j \in \{0,1,\ldots,J\}$ on the domain $(T,J) \in \text{dom } \phi$. A hybrid arc ϕ is a solution to the hybrid system $\mathcal{H}$ with input $u \in \mathbb{R}^m$ if the following conditions are met:

- $\phi(0,0) \in cl(C) \cup D$
- For all $j \in \mathbb{N}$ such that $I^j := \{t : (t,j) \in \text{dom } \phi\}$ has nonempty interior, $(\phi(t,j), u(t)) \in C$ for all $t \in \text{int } I^j$ and $\phi(t,j) \in F(\phi(t,j), u(t))$ for almost all $t \in I^j$
- For all $(t,j) \in \text{dom } \phi$ such that $(t, j+1) \in \text{dom}\phi$, $(\phi(t,j), u(t)) \in D$ and $\phi(t, j+1) \in G(\phi(t,j), u(t))$

where $cl(S)$ is the closure of the set S.

The finite-horizon reachability map for a hybrid system $\mathcal{H}$ collects states the system can reach from some initial state ξ over a given time interval $[0, T]$ with at most J jumps. This reachability map is defined in [8] as

$$\mathfrak{R}_{\mathcal{H}}(T, J, \xi) := \{\phi(t, j) : \phi \in \hat{\mathcal{S}}_{\mathcal{H}}(\xi), (t, j) \in \text{dom } \phi \cap \mathfrak{T}(T, J)\} \tag{1.2}$$

with $\hat{\mathcal{S}}_{\mathcal{H}}(\xi)$ denoting the set of all solutions to the hybrid system $\mathcal{H}$ from initial state ξ and $\mathfrak{T}(T, J)$ denoting the hybrid time horizon $[0, T] \times \{0, 1, 2 \dots J\}$.

Notation: The 2 norm of a vector x is denoted as $|x|$, with the infinity norm denoted as $|x|_\infty$. The set of real numbers is represented by $\mathbb{R}$. The subset Y of X is given as $Y \subset X$. The unit ball $\mathbb{B} \subset \mathbb{R}^3$ is defined as $\mathbb{B} := \{x \in \mathbb{R}^3 : |x| \leq 1\}$. The set of natural numbers is denoted as $\mathbb{N}$. The map f of x to y is denoted as $f : x \mapsto y$. The set of unit quaternions $\mathbb{S} \subset \mathbb{R}^4$ is defined as $\mathbb{S} := \{q \in \mathbb{R}^4 : |q| = 1\}$. The vector e_n contains all zeros except the n^{th} position which is 1. The distance from a vector p to a set is defined as $\text{dist}(p, Q) := \{\min(|p^\top q|_\infty) : q \in Q\}$. The derivative with respect to vector x is defined as $\mathcal{D}_x(F) = \frac{\partial vec(F)}{\partial vec(x)^\top}$ with $vec(x)$ denoting the matrix x reshaped as a column vector. The determinant of a matrix R is denoted $det(R)$. For a vector $q \in \mathbb{R}^n$ and a set $P \in \mathbb{R}^n$, the $\arg\max_{p \in P} pq$ is the largest value of pq for all $p \in P$. A sequence of variables x indexed by k is denoted as $\{x_k\}_{k \in \mathbb{N}_{\geq 1}}$. The Minkowski sum is the addition of two sets such that $A + B := \{a + b : a \in A, b \in B\}$

Chapter 2

Problem Statement and Outline of Proposed Solution

The goal of this paper is to provide an algorithm that generates a reference trajectory that steers a quadrotor to a desired hovering location while evading dynamic obstacles with limited information. To formulate the problem, consider a quadrotor $\mathcal{V}$ whose state is x_a and multiple obstacles $\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_N$. Let each obstacle $\mathcal{O}_i$ have state x_{o_i} consisting of position $p_{o_i} \in \mathbb{R}^3$, velocity $v_{o_i} \in \mathbb{R}^3$, and angular velocity $\omega_{o_i} \in \mathbb{R}^3$, with only the obstacle positions known. Consider a target set $\mathbb{T}$ containing all the desired points for the quadrotor position p_a to converge to. Using a feedback control algorithm, the reference trajectory has to steer the quadrotor position to the target set in finite time while avoiding collisions with the obstacles. This is shown in Figure 2.1 with the quadrotor $\mathcal{V}$ moving to reach the target set $\mathbb{T}$ while avoiding obstacles $\mathcal{O}_1$ and $\mathcal{O}_2$.

To solve this problem, we propose a feedback motion-planning algorithm that generates a reference trajectory by computing and manipulating two sets, one describing where the quadrotor can move and the other containing all possible obstacle locations. More precisely, the first set is called mobility set, denoted

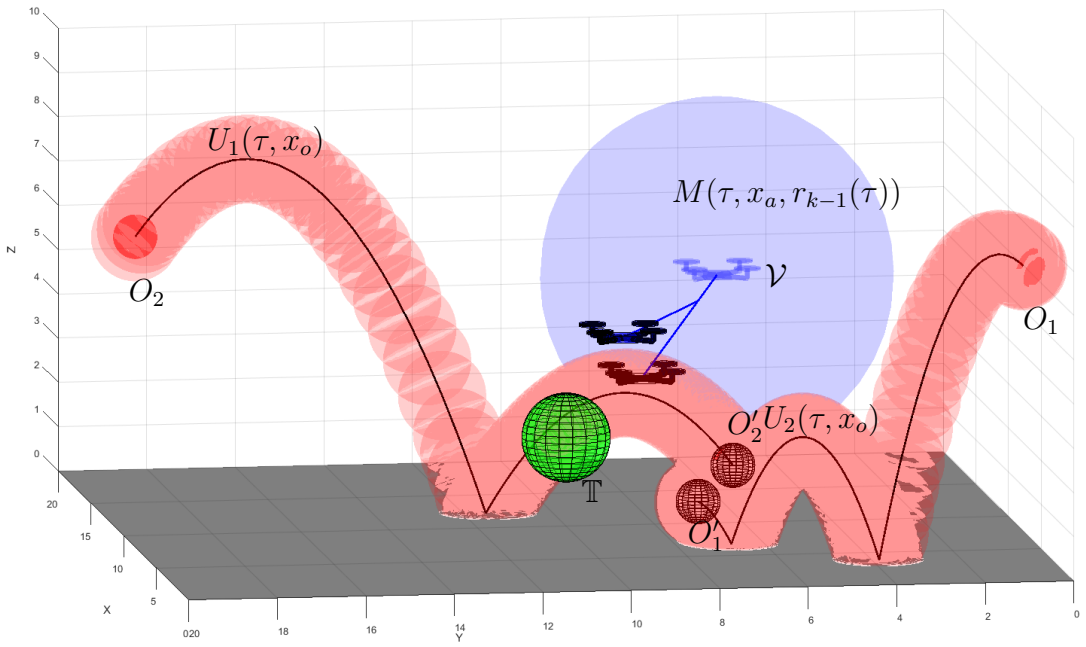

Figure 2.1: The quadrotor $\mathcal{V}$ is moving towards the target $\mathbb{T}$ while avoiding the obstacles $\mathcal{O}_1$ and $\mathcal{O}_2$. The set $M(\tau, x_a, r_{k-1}(\tau))$ is the sphere where the quadrotor can move in τ time from initial state x_a with initial reference state $r_{k-1}(\tau)$. The initial states of obstacles $\mathcal{O}_1$ and $\mathcal{O}_2$ are given by x_o, with $\mathcal{O}_1'$ and $\mathcal{O}_2'$ being the obstacles at time τ. The sets $U_1(\tau, x_o)$ and $U_2(\tau, x_o)$ contain all the unsafe points around the path of $\mathcal{O}_1$ to $\mathcal{O}_1'$ and $\mathcal{O}_2$ to $\mathcal{O}_2'$ over the time period $[0, \tau]$. Moving directly towards the target causes the quadrotor to collide with $\mathcal{O}_2$ at $(10, 8.5, 3)$. The motion planner moves the quadrotor to $(10, 9, 4)$ which avoids the obstacles while moving to the target.

$M(\tau, \xi_a, \xi_r)$ and contains all trajectories that the quadrotor can reach from an initial vehicle state ξ_a and initial reference state ξ_r over the continuous time interval $[0, \tau]$, as shown by $M(\tau, x_a, r_{k-1}(\tau))$ in Figure 2.1. The second set is called the unsafe set, denoted $U_i(\tau, \xi_o)$, contains all points the i^{th} obstacle can reach from initial obstacle state ξ_o over the time interval $[0, \tau]$. In Figure 2.1, the unsafe set for obstacles $\mathcal{O}_1$ and $\mathcal{O}_2$ are $U_1(\tau, x_{o_1})$ and $U_2(\tau, x_{o_2})$. With $M(\tau, \xi_a, \xi_r)$ and $U_i(\tau, \xi_o)$ for each i, the proposed algorithm generates a reference trajectory that, using the reference tracking controller, drives the quadrotor to the target set without colliding with any obstacle.

Chapter 3

Modeling the Obstacles and the Vehicle

In order to define the mobility and unsafe sets, the quadrotor and obstacle models used to generate the sets must be defined.

3.1 Modeling the Obstacles

Each obstacle is considered a bouncing ball moving in $\mathbb{R}^3$, with the i^{th} obstacle's state given by

$$x_{o_i} = (p_{o_i}, v_{o_i}) \tag{3.1}$$

$$x_{o_i} \in \mathcal{X}_{o_i} := \mathbb{R}^6 \tag{3.2}$$

where $p_{o_i} = (p_{x_{o_i}}, p_{y_{o_i}}, p_{z_{o_i}}) \in \mathbb{R}^3$ is position and $v_{o_i} = (v_{x_{o_i}}, v_{y_{o_i}}, v_{z_{o_i}}) \in \mathbb{R}^3$ is velocity. When moving in free air, the motion of a bouncing ball is modeled using

the equations of projectile motion without air resistance, namely

$$\dot{p}_{o_i} = v_{o_i} \tag{3.3}$$

$$\dot{v}_{o_i} = \begin{bmatrix} 0 & 0 & -\gamma \end{bmatrix} \tag{3.4}$$

where $\gamma > 0$ is the gravity acceleration.

The obstacle's impact with the ground is modeled as an instantaneous velocity change, leading to the definition of the jump map $G_{o_i}(x_{o_i})$. Note that this reverses the sign of the vertical velocity and the restitution coefficient $\lambda \in (0, 1)$ models the energy loss at impacts. Accelerations in the x and y directions are generated by the spin of the ball. Since the angular velocities are not known, the exact trajectory cannot be predicted. Instead of a single velocity after a jump, we define the set of possible velocities. This is done to capture the effect of all values withing the expected angular velocity range. The set $\Sigma \subset \mathbb{R}$ contains all possible changes in linear velocity from the expected angular velocities of the ball. The impact can then be modeled as:

$$\begin{aligned} x_{o_i}^+ \in G_{o_i}(x_{o_i}) := \{(p_{x_{o_i}}^+, p_{y_{o_i}}^+, p_{z_{o_i}}^+, v_{x_{o_i}}^+, v_{y_{o_i}}^+, v_{z_{o_i}}^+) : p_{x_{o_i}}^+ = p_{x_{o_i}}, p_{y_{o_i}}^+ = p_{y_{o_i}}, \\ p_{z_{o_i}}^+ = 0, v_{x_{o_i}}^+ = v_{x_{o_i}} + \Sigma, v_{y_{o_i}}^+ = v_{y_{o_i}} + \Sigma, v_{z_{o_i}}^+ = -\lambda v_{z_{o_i}}\} \end{aligned} \tag{3.5}$$

where $+$ is the Minkowski sum. The impacts occur when the obstacle reaches the ground and if we assume the obstacle is a point mass, then impacts occur at $p_{z_{o_i}} = 0$.

The combination of the continuous and discrete dynamics of the bouncing ball leads to the following hybrid model for each obstacle $\mathcal{O}_i$:

$$\mathcal{H}_{o_i} = (C_{o_i}, F_{o_i}, D_{o_i}, G_{o_i}), \tag{3.6}$$

where

$$C_{o_i} := \{x_{o_i} \in \mathbb{R}^6 : p_{z_{o_i}} \geq 0\}$$
$$F_{o_i}(x_{o_i}) = \begin{bmatrix} v_{o_i} & 0 & 0 & -\gamma \end{bmatrix}^\top$$
$$D_{o_i} := \{x_{o_i} \in \mathbb{R}^6 : p_{z_{o_i}} = 0, v_{z_{o_i}} \leq 0\}$$
$$G_{o_i}(x_{o_i}) \in (p_{x_{o_i}}, p_{y_{o_i}}, 0, v_{x_{o_i}} + \Sigma, v_{y_{o_i}} + \Sigma, -\lambda v_{z_{o_i}})$$

3.2 Modeling the Quadrotor and Tracking Controller

The quadrotor $\mathcal{V}$ is modeled after the quadrotor with reference tracking feedback controller proposed in [2]. The quadrotor state is given as

$$x_b = (p_a, v_a, R_a, \omega_a) \tag{3.7}$$

where $p_a \in \mathbb{R}^3$ is position in world inertial frame, $v_a \in \mathbb{R}^3$ is linear velocity in world inertial frame, $R_a \in SO(3)$ with $SO(3) = \{R \in \mathbb{R}^{3\times3} : R^\top R = I_3, det(R) = 1\}$ is rotation with respect to the world frame as a rotation matrix, and $\omega_a \in \mathbb{R}^3$ is the angular velocity with respect to the world frame. The dynamics of the quadrotor are

$$\dot{p}_a = v_a \tag{3.8}$$
$$\dot{v}_a = -R_a e_3 \frac{f}{m} + g e_3 \tag{3.9}$$
$$\dot{R}_a = R_a S(\omega_a) \tag{3.10}$$
$$\dot{\omega}_a = -J^{-1} S(\omega_a) J \omega_a + J^{-1} \mathcal{M} \tag{3.11}$$

where $e_3 = \begin{bmatrix} 0, 0, 1 \end{bmatrix}^\top$, m is the quadrotor mass, $f \in \mathbb{R}$ is the thrust input, $\mathcal{M} \in \mathbb{R}^3$ is the torque input, $J \in \mathbb{R}^{3\times 3}$ is the inertia tensor, and $S(x)$ is the skew-symmetric matrix form of some vector x, namely

$$S(x) = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}$$

The hybrid tracking controller is comprised of two parts. The first is a saturation controller which reduces the translational error between the state and the reference. The second uses the output of the saturation controller with the current state and a memory variable to drive the quadrotor rotation to the reference. The controller is captured by the hybrid system $\mathcal{H}_a(\xi_a, r) = (C_c, F_c, D_c, G_c)$ with the data defined in (A.14)-(A.17). More detailed information can be found in the appendix.

The combined quadrotor and tracking controller is modeled by the hybrid system

$$\mathcal{H}_a = (C_a, F_a, D_a, G_a) \tag{3.12}$$

with state

$$x_a := (x_b, x_c) \tag{3.13}$$

$$x_a \in \mathcal{X}_a := \mathbb{R}^6 \times SO(3) \times \mathbb{R}^6 \times \{-1, 1\} \times \mathbb{S}^3 \tag{3.14}$$

for a given initial state ξ_a and reference trajectory $r : \mathbb{R}_{\geq 0} \to \mathbb{R}^{12} \times SO(3) \times \mathbb{R}^3, T \in \mathbb{R}_{\geq 0}$ with

$$r(t) := (p_r(t), p_r^{(1)}(t), p_r^{(2)}(t), p_r^{(3)}(t), R_r(t), \omega_r(t)), \tag{3.15}$$

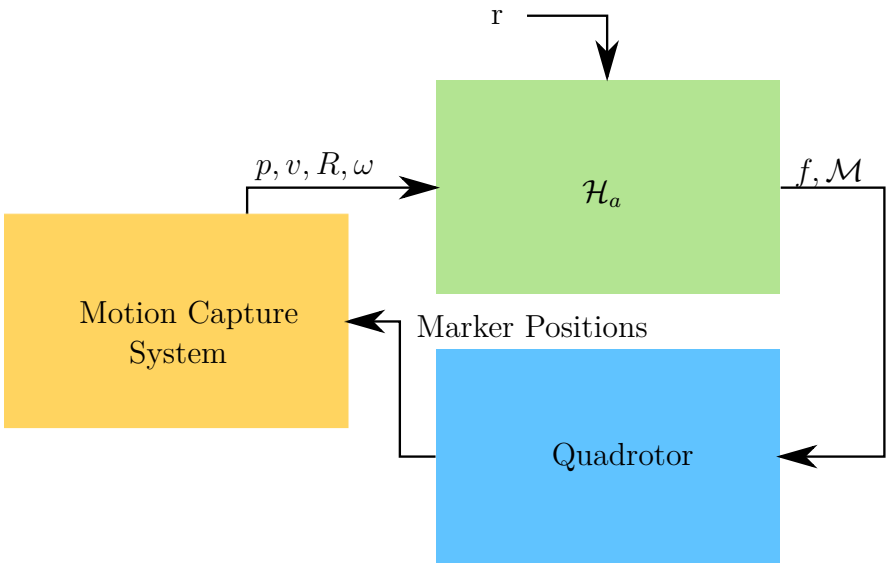

Figure 3.1: Diagram of the feedback loop. The hybrid controller generates a torque and thrust for the quadrotor from the current state and reference trajectory. The motion capture system collects the position, velocity, rotation, and angular velocity of the quadrotor and updates the state of the quadrotor within the hybrid controller.

satisfying the following assumption.

Assumption 1. *Given $\mathcal{M}_p, \mathcal{M}_\omega > 0$, the reference trajectory $t \mapsto r(t)$ is a solution to*

$$\dot{r} \in F_d(r) := (p_r^{(1)}, p_r^{(2)}, p^{(3)}\_r, \mathcal{M}_p\mathbb{B}, R_r S(\omega_r), \mathcal{M}_\omega\mathbb{B}), \tag{3.16}$$

such that rge $r \in \Omega_r$ *for some compact set*

$$\Omega_r \subset \mathbb{R}^{12} \times SO(3) \times \mathbb{R}^3, \tag{3.17}$$

satisfying $e_3^\top R_r(t) e_3 \geq 0$ for each $t \geq 0$ (see [2]).

The data for the hybrid system $\mathcal{H}_a(\xi_a, r(t)) = (C_a, F_a, D_a, G_a)$ modeling the

closed-loop controller and quadrotor dynamics is:

$$C_a := C_c \tag{3.18}$$

$$F_a(x_a, r(t)) := \Big(\dot{p}_a, \dot{v}_a, \dot{R}_a, \dot{\omega}_a, F_c(x_a, r(t))\Big) \tag{3.19}$$

$$D_a := D_c \tag{3.20}$$

$$G_a(x_a, r(t)) := \begin{bmatrix} x_b \\ G_c(x_a, r(t)) \end{bmatrix} \tag{3.21}$$

Chapter 4

Mathematical problem statement

The problem outlined in Chapter 2 can be formulated mathematically as

Problem 1. *Given a quadrotor $\mathcal{V}$ with reference tracking controller whose dynamics are captured by $\mathcal{H}_a$, initial quadrotor state ξ_a, obstacles $\mathcal{O}_1, \mathcal{O}_2, \ldots \mathcal{O}_N$ whose dynamics are captured by $\mathcal{H}_o$, initial obstacle states ξ_{o_i} for $i \in \{1, 2, \ldots, N\}$, minimum safe quadrotor obstacle distance k_u, target closed set $\mathbb{T} \subset \mathbb{R}^3$, and duration $\tau \in \mathbb{R}_{>0}$, compute a reference trajectory given by the sequence $\{r_k\}_{k \in \mathbb{N}_{\geq 1}}$ such that*

1. $r_1(0) = \xi_a$
2. *For each $k \in \mathbb{N}_{\geq 1}$, with x_a and x_{o_i} being the current quadrotor and obstacle states at time $t_c = (k-1)\tau$*
   - *(a) r_{k-1} has been executed by the quadrotor for τ time*
   - *(b) $r_k(t_c) = r_{k-1}(t_c)$*
   - *(c) $[t_c, t_c + \tau] \ni t \mapsto r_k(t)$ satisfies Assumption 1*
   - *(d) The maximal solution ϕ_k to $\mathcal{H}_a$ from x_a for input r_k evolves for at least τ seconds of flow and satisfies $\operatorname{dist}\left(\phi_k(t,j), \mathfrak{R}_{\mathcal{H}_{o_i}}(\tau, \infty, x_{o_i})\right) \geq k_u$ for all $(t,j) \in \operatorname{dom}\phi_k \cap ([t_c, t_c + \tau] \times \mathbb{N})$ so the distance between any possible*

obstacle trajectory and the quadrotor is never below the minimum safe distance

3. *The maximal solution ϕ to $\mathcal{H}_a$ from ξ_a for input r, with r being the concatenation of all reference trajectories in $\{r_k\}_{k\in\mathbb{N}}$, satisfies*

   (a) *ϕ is complete*

   (b) *There exists a finite time t_f such that $p_\phi(t,j) \in \mathbb{T}$ for all $(t,j) \in \text{dom } \phi$ such that $t \geq t_f$.*

In the above problem, $\mathcal{H}_a$ is the feedback control algorithm mentioned in Chapter 2 which is responsible for steering the quadrotor. The evasion of obstacles is accomplished by requirement 2d on each trajectory r_k, enforcing a minimum distance between the obstacle and quadrotor. Convergence of the quadrotor to the target set within some finite time is included by requirement 3b.

Chapter 5

The Motion-Planning Algorithm

To solve the Problem 1, we propose a feedback motion planning algorithm. The algorithm operates over multiple iterations, with each iteration extending the executed reference trajectory by τ_e ordinary time. To prevent the planner from selecting a trajectory from which there is no safe extension due to an obstacle outside of τ_e horizon, each iteration plans for $\tau_p > \tau_e$ ordinary time. The proposed algorithm is as follows:

Step 1: Compute quadrotor mobility set for the quadrotor model in (3.18)-(3.21).

Step 2: Compute unsafe set for all obstacles using the model in (3.6).

Step 3: Remove trajectories from the mobility set $M(\tau, x_a, r_{(k-1)})$ that violate safety constraints, building the safe mobility set.

Step 4: Solve an optimization problem selecting the lowest cost reference trajectory from the safe mobility set.

Step 5: Execute τ_e seconds of the reference trajectory.

Step 6: Go to Step 1 to replan from the new quadrotor and obstacle states.

5.1 Supporting Sets and Constraints

The quadrotor mobility set is the set of all possible reference trajectory quadrotor trajectory pairs (r, ϕ). To generate this set, all possible reference trajectories must be generated. Since this is difficult to implement for most systems in practice, we approximate the set using a motion primitive library Ω_p. The library is generated using the quadrotor dynamics in (3.8)-(3.11) by using inputs f and $\mathcal{M}$ satisfying

$$-\dot{\omega} e_3 \frac{f}{m} - 2\left(RS(\omega)e_3\frac{\dot{f}}{m}\right) - Re_3\frac{\ddot{f}}{m} \in \mathcal{M}_p\mathbb{B}$$

$$\mathcal{M} \in \mathcal{M}_\omega\mathbb{B}$$

with $\mathcal{M}_p$ and $\mathcal{M}_\omega$ being the quadrotor's maximum snap and angular acceleration respectively. The restrictions on f and $\mathcal{M}$ are to ensure all trajectories within Ω_r satisfy the conditions of Assumption 1. These reference trajectories are used as inputs to simulations of the closed-loop quadrotor model (3.12) from the current quadrotor state, with the resulting trajectories building the solution set $\hat{\mathcal{S}}_{\mathcal{H}_a}(\xi_a)$. The quadrotor mobility set $M(\tau, \xi_a, \xi_r)$ containing all reference trajectory quadrotor trajectory pairs (r, ϕ) for the given initial vehicle state ξ_a, motion primitive library Ω_p, and initial reference state ξ_r over the time interval $[0, \tau]$, is defined as

$$M(\tau, \xi_a, \xi_r) := \Big\{(r,\phi) : \phi(t,j) = \phi_a(t,j), \forall (t,j) \in \text{dom } \phi \cap [0,\tau] \times \mathbb{N}, \\ \phi_a \in \hat{\mathcal{S}}_{\mathcal{H}_a}(\xi_a, r), \forall r \in \Omega_p, r(0) = \xi_r\Big\} \quad (5.1)$$

where the set $\hat{\mathcal{S}}_{\mathcal{H}_a}(\xi_a, r)$ contains all solutions to the hybrid system $\mathcal{H}_a$ from initial state ξ_a for reference trajectory r.

The set of all possible obstacles states is called the unsafe set. For each obstacle, the unsafe set is denoted as $U_i(\tau, \xi_{o_i})$ for the given initial state ξ_{o_i} over the

hybrid time horizon $[0, \tau] \times \mathbb{N}$. The set is defined as

$$U_i(\tau, \xi_{o_i}) := \mathfrak{R}_{\mathcal{H}_{o_i}}(\tau, \infty, \xi_{o_i}). \tag{5.2}$$

The set containing all possible states of all obstacles is the unsafe set $U(\tau, \xi_o)$, defined as

$$U(\tau, \xi_o) := \bigcup_{i=1}^{N} U_i(\tau, \xi_{o_i}). \tag{5.3}$$

The safe mobility set, $M_S(\tau, \xi_a, \xi_r, \xi_o)$ is constructed by removing any trajectories in $M(\tau, \xi_a, \xi_r)$ which violate the minimum safe distance to the unsafe set $U(\tau, \xi_o)$.

$$M_S(\tau, \xi_a, \xi_r, \xi_o) = \Big\{ (r, \phi_a) \in M(\tau, \xi_a, \xi_r) : \text{dist}(p_{\phi_a}(t_a, j_a), p_{\phi_o}(t_o, j_o)) \geq k_u, \\ \forall (t_a, j_a) \in \text{dom } \phi_a \text{ and } \forall (t_o, j_o) \in \text{dom } \phi_o, \ \textit{for all } \phi_o \in U(\tau, \xi_o) \Big\} \tag{5.4}$$

where $p_{\phi_a}(t, j)$ is the positional component of $\phi(t, j)$. The constant k_u is the constraint on the minimum allowed distance between the quadrotor position and the unsafe set for a trajectory to be considered safe.

5.2 Problem Reformulation and Algorithm

Using the above sets, each reference trajectory from Problem 1 can be restated as follows:

Problem 2. *Given $\mathbb{T} \subset \mathbb{R}^3$, planning window $\tau_p \in \mathbb{R}_{>0}$, execution window $\tau_e \in (0, \tau_p]$, previous reference trajectory $r_{prev} : [0, \tau_p] \to \Omega_r$, previous reference cost κ_{prev}, vehicle state $x_a \in \mathcal{X}_a$, and obstacle state $x_o \in \mathcal{X}_o$, generate a reference trajectory $\hat{r} \in \Omega_p$ with domain $[0, \tau_p]$ which will*

$$\underset{\hat{r}}{\text{minimize}}\ \kappa(\hat{r}, \phi, r_{prev}, \kappa_{prev})$$

subject to

C1) $\hat{r}(0) = r_{prev}(\tau_e)$

C2) $(\hat{r}, \phi) \in M_S(\tau_p, x_a, r_{prev}, x_o)$

where τ_p *is the planning window and* ϕ *is the solution to* $\mathcal{H}_a$ *from initial state* x_a *with reference trajectory* $\hat{r}$ *over* $[0, \tau_p] \times \mathbb{N}$.

The cost functional κ is defined as

$$\kappa(\hat{r}, \phi, r_{prev}, \kappa_{prev}) := \text{dist}(p_\phi(\tau_p, j_\tau), \mathbb{T}) + \kappa_p(\hat{r}, r_{prev}, \kappa_{prev}) \tag{5.5}$$

where

$$\kappa_p(\hat{r}, r_{prev}, \kappa_{prev}) := \begin{cases} 0 & \text{if } \hat{r}(t) = r_{prev}(t + \tau_e), for\ all\ t \in [0, \tau_p - \tau_e] \\ k_h \kappa_{prev} & otherwise \end{cases}$$

with j_τ being the number of jumps at time τ_e, $p_\phi(t, j)$ denoting the position of the trajectory ϕ at (t, j), and $k_h \in \mathbb{R}_{\geq 0}$ being the hysteresis tuning constant.

Chattering is a scenario where a planner switches between different trajectories without making progress to the goal. The planner can chatter when the quadrotor and target are on opposite sides of an obstacle and there is noise in the quadrotor position. This will result in the planner changing the path around the obstacle at each iteration, resulting in the quadrotor becoming trapped. To prevent this, a hysteresis term κ_p is introduced that increases the cost of references which do not share the first $\tau_p - \tau_e$ seconds of trajectory with the last $\tau_p - \tau_e$ seconds of the previous reference trajectory. This punishes switching trajectories, reducing chattering.

Solving Problem 2 for each reference while the previous is followed by the quadrotor, results in Algorithm 1.

Algorithm 1: Motion planning algorithm with input $(\mathbb{T}, \tau_p, \tau_e, \xi_a, \xi_o, \mathcal{H}_a, \mathcal{H}_{o_i})$

$\kappa_0 \leftarrow 0$
$r_0 \leftarrow 0$
$x_a \leftarrow \xi_a$
$x_o \leftarrow \xi_o$
for $k = 1, 2, \ldots$ **do**
  $M \leftarrow \emptyset$
  $U \leftarrow \emptyset$
  $M_S \leftarrow \emptyset$
  for all $\hat{r} \in \Omega_p, \hat{r}(0) = r_{k-1}(\tau_e)$ **do**
    The solution ϕ_a of $\mathcal{H}_a$ is simulated from x_a for reference $\hat{r}$ for $[0, \tau_p]$ seconds of flow
    $M \leftarrow M \cup \{(\hat{r}, \phi_a)\}$
  end for
  for $i \in \{1, 2, \ldots, N\}$ **do**
    The solution ϕ_o of $\mathcal{H}_{o_i}$ is simulated from x_{o_i} for $[0, \tau_p]$ seconds of flow
    $U \leftarrow U \cup \{\phi_o\}$
  end for
  for all $(\hat{r}, \phi_a) \in M$ **do**
    for all $\phi_o \in U$) **do**
      if $\text{dist}(p_{\phi_a}(t_a, j_a), p_{\phi_o}(t_o, j_o)) \geq k_u, \forall (t_a, j_a) \in \text{dom } \phi_a$ and $\forall (t_o, j_o) \in \text{dom } \phi_o$ **then**
        $M_S \leftarrow M_S \cup \{(\hat{r}, \phi_a)\}$
      end if
    end for
  end for
  (r_k, ϕ) is the solution to Problem 2 given $(\mathbb{T}, \tau_p, \tau_e, r_{k-1}, \kappa_{k-1}, x_a, x_o)$
  $\kappa_k \leftarrow \kappa(r_k, \phi, r_{k-1}, \kappa_{k-1})$
  Execute r_k for τ_e seconds.
  Update x_a and x_o
end for

Chapter 6

Simulations

Simulations have been run using both a double integrator point mass model and the combined quadrotor controller model. For the simulation in Figure 6.1, the point mass model was used. The mobility set was discritzed at by applying 1000 different input acceleration vectors to the model and simulating the behavior from its current state for 0.5 seconds. The acceleration vectors were generated using five different acceleration values in steps of 4.6 at 20 evenly spaced angles from 0 to 1.9π in the xy plane and 10 evenly spaced angles from -0.5π to 0.5π in the xz plane. By checking for collision with the simulated obstacle set the unsafe trajectories were removed and the remaining trajectories build the set $M_S(\tau, x_a, x_o, r_{k-1})$.

For the simulation shown in Figure 6.2, the hybrid controller and quadrotor dynamical model was used. The set of possible reference trajectories was approximated using 1728 different reference trajectories, generated from 12 increments of each pitch torque, roll torque, and thrust as inputs to (3.8)-(3.11). All reference trajectories had no yaw torque. The planning time was $\tau_p = 0.3$ seconds and execution time of $\tau_e = 0.05$ seconds. The mass, inertial tensor, and single motor maximum thrust were from the system identification of the Crazyflie quadrotor in [4]. The controller constants were the same as in the simulations of [2], except

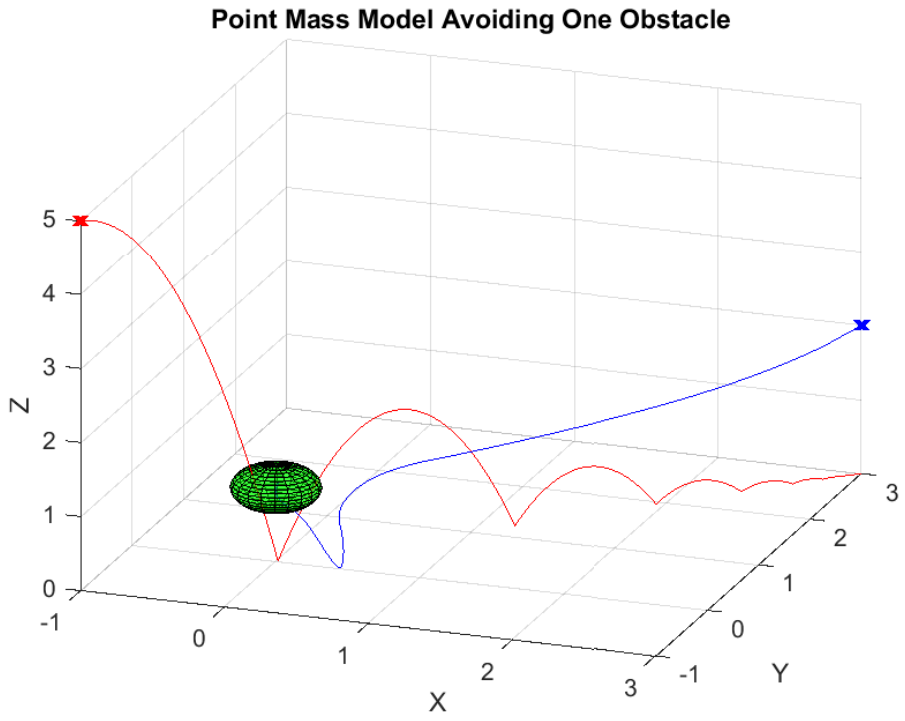

Figure 6.1: Simulation using a point mass model. The quadrotor moves directly toward the target until the obstacle is within the planning window around $(0.3, 0.3, 1.5)$. The quadrotor then makes a left turn to avoid the bouncing obstacle before returning to moving toward the target set. Initial quadrotor location is denoted as the blue x at $(3, 3, 2)$ and initial obstacle location is marked by the red x at $(-1, -1, 5)$ and had an initial velocity of $(1, 1, 0)$. The planning period was $0.5s$, with an execute period $0.2s$, and an obstacle radius of $0.3m$. The target set was a sphere centered at $(0, 0, 1)$ with a radius of $0.3m$. The maximum acceleration of the point mass was $23m/s^2$.

$\alpha = 0.5, \delta = 0.5$ and $\beta = k_v/4$. The code for the simulations can be found at `https://github.com/HybridSystemsLab/CrazyFlieAvoidanceSimulation`.

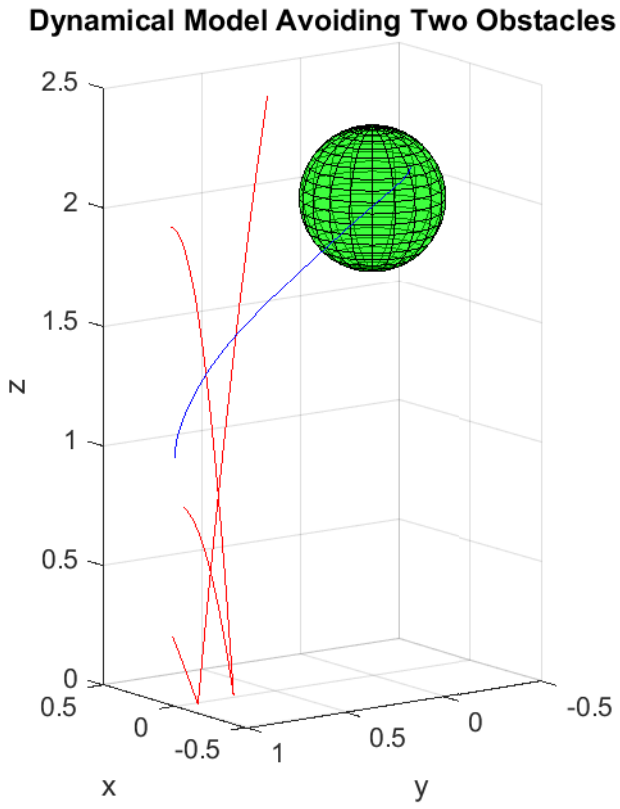

Figure 6.2: Simulation using quadrotor dynamical model. The quadrotor moves to the negative x direction to avoid the two obstacles then arcs back toward the target set. Initial quadrotor state is $(0, 1, 3, 0, 0, 0, I_{3\times3}, 0, 0, 0)$ and initial obstacle states $(-0.03, 0, 4, 0, 1, 0)$ and $(0.03, 1, 2, 0, -0.5, 0)$. The planning period was 0.3 seconds, with an execute period $0.05s$, and an obstacle radius of $0.05m$. The target set was a sphere centered at $(0, 0, 4)$ with a radius of $0.3m$.

Chapter 7

Experiments

To show the algorithm's viability for use in online motion planning scenarios with limited compute resources the following experiment was conducted.

7.1 Experimental Setup

The experiments were run on a Windows 10 computer with a dual core $3.20GHz$ processor with 8GB of memory. Quadrotor and obstacle position data was captured using eight motion capture cameras running at $120Hz$. Velocities were calculated using the difference in position and capture time between two frames. The experimental quadrotor was a Crazyflie 2.0 controlled over $2.4GHz$ radio though the Crazyflie client. The obstacles used were $0.08m$ diameter wiffle balls wrapped in retro-reflective tape to allow tracking by the motion capture system. To recover the restitution constant and Σ factor, one obstacle was tossed 15 times with different spins from a height of approximately $1.5m$. The restitution constant was calculated as the average change in vertical velocity from before to after the impact. The set Σ was constructed using the largest change in horizontal velocity from before to after the impact. The ball was found to have a restitution constant

of $\lambda = 0.65$ and possible velocity change from spin of $\Sigma = [-0.02, 0.02] m/s$. The obstacles were thrown toward the quadrotor from a horizontal distance of $0.15m$ to $0.7m$ and a height between $0.7m$ and $0.85m$ with staring horizontal velocities between $0m/s$ and $0.01m/s$ and vertical velocities between $-0.2m/s$ and $0.6m/s$. The motion planner and controller were implemented in Matlab. The experimental quadrotor controller was comprised of four PIDs which drove the quadrotor to the reference. The PIDs output desired thrust, yaw, pitch, and roll values which were sent to the Crazyflie client over ZeroMQ at a rate of $50Hz$. The motion planner used a planning window of $\tau_p = 0.3s$, execution window of $\tau_e = 0.28s$, and an minimum safe distance of $k_u = 0.2m$.

7.2 Experimental Results

The quadrotor was able to reliably avoid the obstacles and converge to the target set in each test. The motion planner implemented had low computational delay with planner updates having a mean computation time of $83ms$, median of $84ms$, and a maximum time of $138ms$. In figure 7.2, the distance between the quadrotor and the obstacle of multiple experiments are shown, with the minimum being $0.16m$. The violations of the minimum unsafe distance were due to the communication delay between the Crazyflie and the controller, which were not accounted for by the controller. Code for the experiments can be accessed at `https://github.com/HybridSystemsLab/CrazyFlieAvoidanceExperiment`.

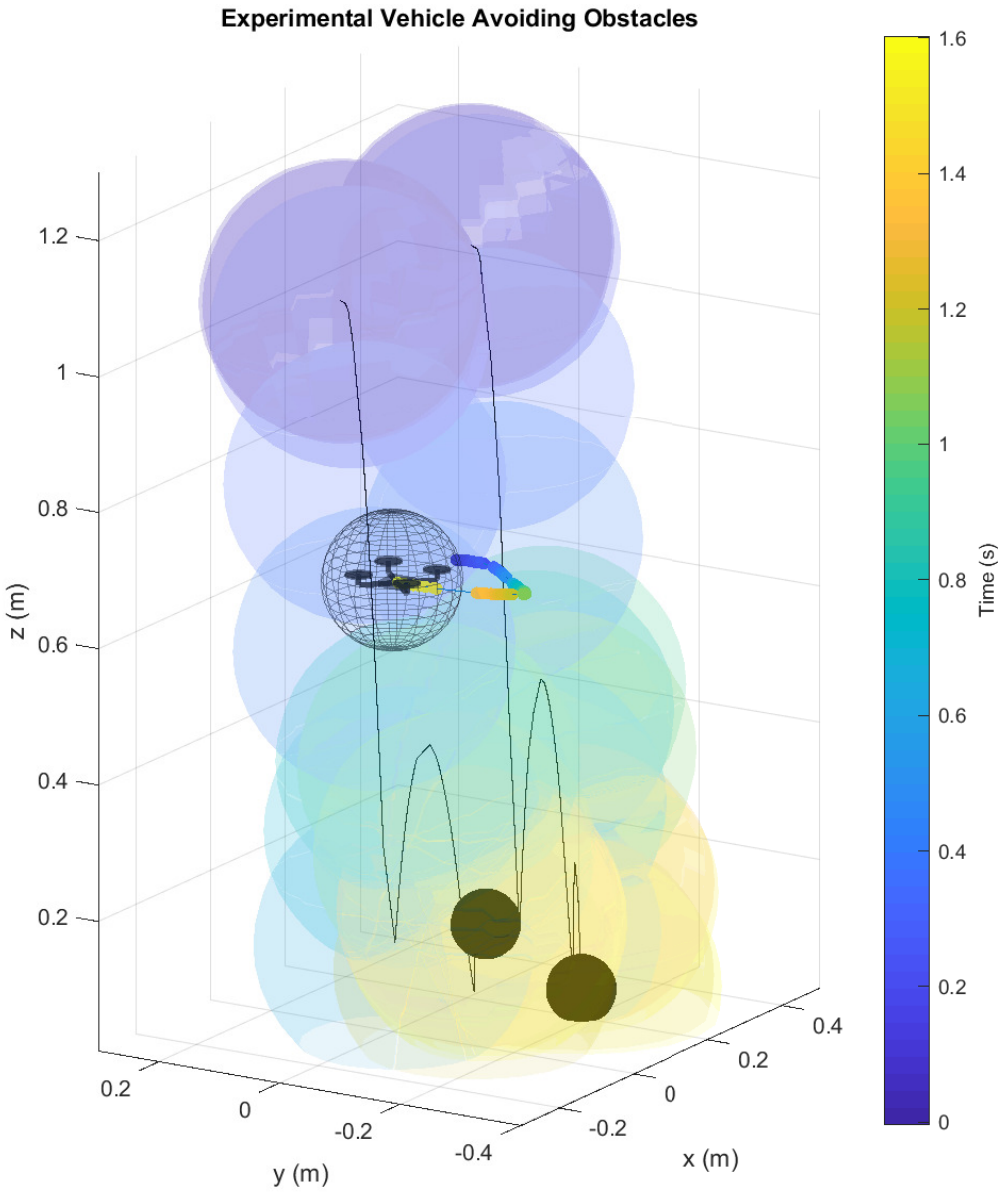

Figure 7.1: Plot of the Crazyflie avoiding two obstacles. The Crazyflie has an initial position of $(0.29, -0.20, 0.68)$ and the obstacles had initial positions of $(0.44, -0.11, 1.08)$ and $(0.43, -0.10, 1.07)$. The light colored spheres depict the unsafe set, with the color of the sphere denoting the time as shown in the color bar. The unsafe set at time t is all spheres between t and $t + \tau_e$. The smaller colored spheres show the position of the Crazyflie over time. The wire frame sphere denotes the target set centered at $(0.18, -0.12, 0.67)$ with radius $0.1m$

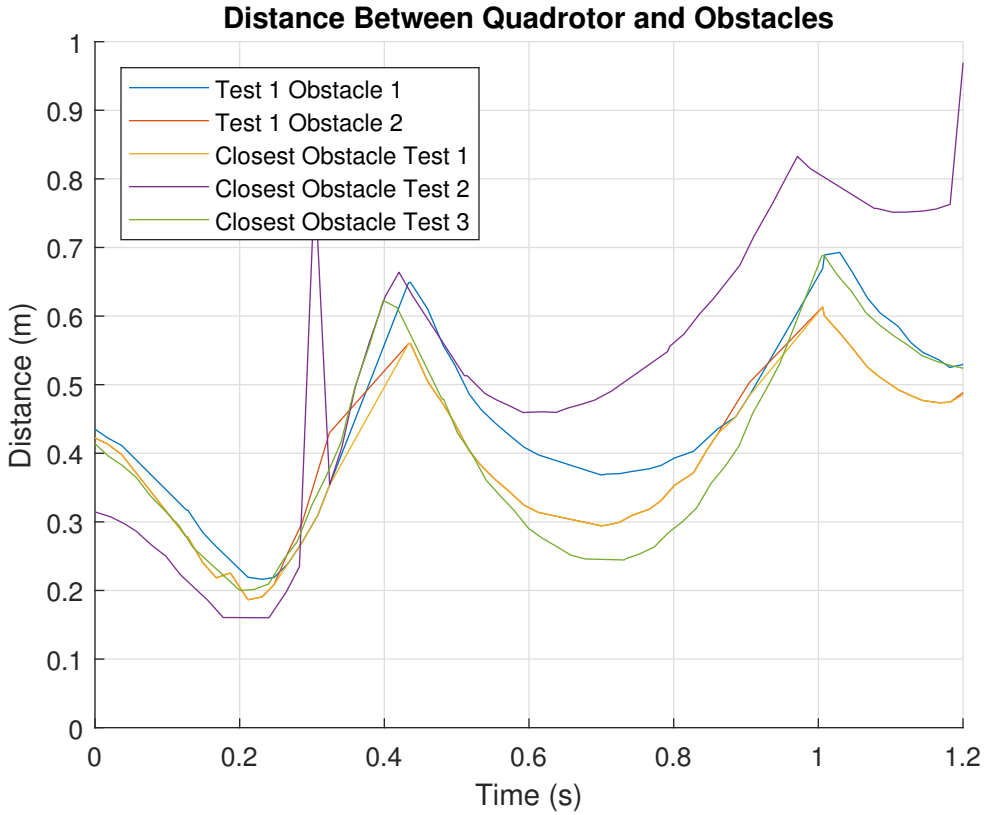

Figure 7.2: Plot of distance between quadrotor and obstacles over time for three experiment runs. In each experiment, the quadrotor was avoiding two obstacles. Large jumps in distance are caused by the camera system not capturing one of the obstacles for a frame, resulting in the further obstacle to become the closest.

Chapter 8

Conclusion

In this paper, we presented a set based feedback motion planner for a quadrotor avoiding bouncing ball like dynamic obstacles with limited obstacle state information. Simulation using the hybrid controller and quadrotor model shows the algorithm's ability to avoid multiple obstacles while converging to the target. Experimental results show the algorithm's feasibility in real world scenarios with limited computing power and real time constraints.

Appendix A

Hybrid Reference Tracking Controller

The hybrid reference tracking feedback controller is composed of two parts. The first minimizes position and velocity tracking errors, p_0 and v_0, and combined tracking error $\widetilde{r}$ for a reference position p_r and velocity v_r. The tracking errors are defined as $p_0 = p_r - p_a$, $v_0 = v_r - v_a$, and $\widetilde{r} = k_p p_0 + k_v v_0$. Within the combined tracking error, the tuning constraints k_p and k_v are used to determine the weight of the position and velocity error. Using these errors, an integral state z is defined within the controller, where

$$\dot{z} = k_z \mathcal{D}_{v_0} (\overline{V}_0(p_0, v_0))^\top \tag{A.1}$$

$$\overline{V}(p_0, v_0) = \sum_{i=1}^{3} \frac{1}{2} \begin{bmatrix} \sigma_K(\widetilde{r}_i) & e_i^\top v_0 \end{bmatrix} \begin{bmatrix} \frac{k_v}{k_p}\beta & -\beta - \beta & k_p \end{bmatrix} \begin{bmatrix} \sigma_K(\widetilde{r}_i) \\ e_i^\top v_0 \end{bmatrix} + \int_0^{r_i} \sigma_K(\xi) d\xi \tag{A.2}$$

with $\beta \in (0, k_v)$. The function σ_K is a K saturation function mapping $\sigma_K : \mathbb{R} \to \mathbb{R}$.

The second part of the controller seeks to drive the rotation of the quadrotor to that of the reference trajectory. Since the rotation of the quadrotor determines its acceleration, the positional controller must change the attitude of the quadrotor to drive the position and velocity to that of the reference. The rotational error $R_1 \in SO(3)$ is therefore defined between the reference trajectory and the rotation determined by the positional controller. The rotation from the translational controller $R_0 \in SO(3)$ is

$$R_0 = \left(I_3 + S\left(-S(R_a e_3)\frac{\mu}{|\mu|} \right) + \frac{S(-S(R_a e_3)\frac{\mu}{|\mu|})^2}{1 - e_3^\top R_a^\top \frac{\mu}{|\mu|}} \right) R_a \tag{A.3}$$

The term μ is the output of the feedback law

$$\mu(r, p_o, v_o, z) = a_r - \begin{bmatrix} \sigma_K(\widetilde{r}_1) \\ \sigma_K(\widetilde{r}_2) \\ \sigma_K(\widetilde{r}_3) \end{bmatrix} - \begin{bmatrix} \sigma_K(z_1) \\ \sigma_K(z_2) \\ \sigma_K(z_3) \end{bmatrix} - g e_3 \tag{A.4}$$

This rotation error is then defined as

$$R_1 := R_a R_0^\top \tag{A.5}$$

From this rotation error, there exists a unique unit quaternion $q_1 = [\eta_1, \epsilon_1] \in \mathbb{S}^3$ with scalar component η_1 and vector component ϵ_1 satisfying

$$R_1 = I_3 + 2\eta_1 S(\epsilon_1) + 2S(\epsilon_1)^2 \tag{A.6}$$

and quadrotor dynamics defined in [2]. This error quaternion satisfying (A.6) and the quadrotor dynamics can also be found by the controller using the memory

variable $\hat{q}_1 \in \mathbb{S}^3$ as follows

$$q_1 := \underset{p \in (R_1)}{\arg\max}\, \hat{q}_1^\top p. \tag{A.7}$$

From this error quaternion q_1, the angular velocity from the translational controller is defined as

$$\omega_0 = -\frac{1}{2} R_0^\top \Gamma(R)_0^\top \mathcal{D}_t(R_0) + R_0^\top(-\omega_0^\star - k_q h \epsilon_1) \tag{A.8}$$

$$\omega_0^\star = \frac{2k_z k_{V_0}}{kh}(\eta_1 S(\mu) - S(\mu)S(\epsilon_1))\mathcal{D}_{v_0}(V_0)^\top \tag{A.9}$$

$$\Gamma(R) = -\begin{bmatrix} S(Re_1) & S(Re_2) & S(Re_3) \end{bmatrix}^\top$$

The variable $h \in \{-1, 1\}$ is a logic variable used by the controller to select the direction of the rotation error. In (A.9), the continuous function $V_0(p_0, v_0, z)$ is defined as

$$V_0(p_0, v_0, z) := k_z \overline{V}(p_0, v_0) + \sum_{i=1}^{3} \left(\int_0^{|z_i|} \sigma_K(\xi) d\xi \right). \tag{A.10}$$

The tuning constraint k_z sets the weight of $\overline{V}(p_0, v_0)$ relative to the integral state.

The torque and thrust passed to the quadrotor from these controllers are given by

$$f = m|\mu| \tag{A.11}$$

and

$$\mathcal{M} = S(\omega_a) J \omega_a + J\Big(- k_\omega(\omega_a - \omega_0) - khR_0^\top \epsilon_1 + \mathcal{D}_t(\omega_0) \Big). \tag{A.12}$$

Combining the above equations, the hybrid reference tracking controller $\mathcal{H}_c = (C_c, F_c, D_c, G_c)$ with state

$$x_c = \begin{bmatrix} z & h & \hat{q}_1 \end{bmatrix}^\top \tag{A.13}$$

and set of possible states $\mathcal{X}_c := \mathbb{R}^3 \times \{-1, 1\} \times \mathbb{S}^3$ has data

$$C_c := \left\{ x_c \in \mathcal{X}_c : \Big(\underset{p \in \mathcal{Q}(R_1)}{\arg\max} (\hat{q}_1^\top p), h \Big) \in Q_\delta^+, \operatorname{dist}(\hat{q}_1, \mathcal{Q}(R_1)) < \alpha \right\} \tag{A.14}$$

$$F_c(x_c, r(t)) := \begin{bmatrix} k_z \mathcal{D}_{v_0} (\overline{V}_0(p_0, v_0))^\top \\ 0 \\ 0 \end{bmatrix} \tag{A.15}$$

$$\begin{aligned} D_c &:= D_1 \cup D_2 \\ D_1 &:= \left\{ x_c \in \mathcal{X}_c : (\underset{p \in \mathcal{Q}(R_1)}{\arg\max} (\hat{q}_1^\top p), h) \in \mathcal{Q}_\delta^- \right\} \\ D_2 &:= \{ x_c \in \mathcal{X}_c : \operatorname{dist}(\hat{q}_1, \mathcal{Q}(R_1)) \geq \alpha \} \end{aligned} \tag{A.16}$$

$$G_c(x_c, r(t)) := \begin{cases} (z, -h, \hat{q}_1) & \text{if } x_c \in D_1 \\ (z, h, \arg\max_{p \in \mathcal{Q}(R_1)} (\hat{q}_1^\top p)) & \text{if } x_c \in D_2 \end{cases} \tag{A.17}$$

where $0 < \alpha < 1$. The sets Q_δ^- and Q_δ^+ are defined as $Q_\delta^- = \{(q, h) \in \mathbb{S}^3 \times H : h\eta \leq -\delta\}$, $Q_\delta^+ = \{(q, h) \in \mathbb{S}^3 \times H : h\eta \geq -\delta\}$ for a constant $\delta \in (0, 1)$. The set $\mathcal{Q}(R)$ contains all quaternions $\{q, -q\} \subset \mathbb{S}^3$ satisfying $R(q) = R(-q) = R$ for each $R \in SO(3)$. For more information, see [2].

Bibliography

[1] Olov Andersson, Oskar Ljungqvist, Mattias Tiger, Daniel Axehill, and Fredrik Heintz. Receding-Horizon Lattice-Based Motion Planning with Dynamic Obstacle Avoidance. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 4467–4474, December 2018. ISSN: 2576-2370.

[2] P. Casau, R. G. Sanfelice, R. Cunha, D. Cabecinhas, and C. Silvestre. Robust global trajectory tracking for a class of underactuated vehicles. *Automatica*, 58:90–98, August 2015.

[3] Cosmin Copot, Andres Hernandez, Thi Thoa Mac, and Robin De Keyse. Collision-free path planning in indoor environment using a quadrotor. In *2016 21st International Conference on Methods and Models in Automation and Robotics (MMAR)*, pages 351–356, August 2016.

[4] Julian Forster. System identification of the crazyflie 2.0 nano quadrocopter. page 147.

[5] R. Goebel, R. G. Sanfelice, and A. R. Teel. *Hybrid Dynamical Systems: Modeling, Stability, and Robustness*. Princeton University Press, New Jersey, 2012.

[6] Björn Lindqvist, Sina Sharif Mansouri, Ali-akbar Agha-mohammadi, and George Nikolakopoulos. Nonlinear MPC for Collision Avoidance and Control of UAVs With Dynamic Obstacles. *IEEE Robotics and Automation Letters*, 5(4):6001–6008, October 2020. Conference Name: IEEE Robotics and Automation Letters.

[7] Sikang Liu, Nikolay Atanasov, Kartik Mohta, and Vijay Kumar. Search-based motion planning for quadrotors using linear quadratic minimum time control. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2872–2879, September 2017. ISSN: 2153-0866.

[8] Mohamed Maghenem and Ricardo G. Sanfelice. Local lipschitzness of reachability maps for hybrid systems with applications to safety. In *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and*

Control, HSCC '20, New York, NY, USA, 2020. Association for Computing Machinery.

[9] Mohamed Mghenem and Ricardo G. Sanfelice. Safety characterization in hybrid inclusions using barrier functions: poster abstract. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pages 282–283, Montreal Quebec Canada, April 2019. ACM.

[10] Julián Rascón-Enríquez, Luis Arturo García-Delgado, José R. Noriega, Alejandro García-Juárez, and Eduardo S. Espinoza. Tracking Control for Quad-Rotor Using Velocity Field and Obstacle Avoidance Based on Hydrodynamics. *Electronics*, 9(2):233, February 2020. Number: 2 Publisher: Multidisciplinary Digital Publishing Institute.