Fall 2020

SHARED CONTROL FOR MOBILE ROBOT OBSTACLE AVOIDANCE

Matthew Glen Westbrook
University of New Hampshire, Durham

SHARED CONTROL FOR MOBILE ROBOT OBSTACLE AVOIDANCE

BY

MATTHEW WESTBROOK

BS, Mechanical Engineering, University of New Hampshire, United States, 2018

THESIS

Submitted to the University of New Hampshire
in Partial Fulfillment of
the Requirements for the Degree of

Master of Science

in

Mechanical Engineering

September 2020

This thesis was examined and approved in partial fulfillment of the requirements for the degree of Master of Science in Mechanical Engineering by:

Thesis Director, Dr. Se Young Yoon,
Assistant Professor of Electrical Engineering

Dr. May-Win L. Thein,
Associate Professor of Mechanical Engineering

Dr. Bingxian Mu,
Assistant Professor of Mechanical Engineering

On August 6, 2020.

Original approval signatures are on file with the University of New Hampshire Graduate School.

TABLE OF CONTENTS

NOMENCLATURE

Symbols

$g = 9.81 \frac{m}{s^2}$	Graviational acceleration constant.

Abbreviations

DOF	Degrees of Freedom
GUUB	Globally Uniformly Ultimately Bounded
IMU	Inertial Measurement Unit
PIV	Proportional Integral Velocity
UUB	Uniformly Ultimately Bounded
VTOL	Vertical Take Off and Landing

LIST OF FIGURES

Figure **Page**

ABSTRACT

Shared Control for Mobile Robot Obstacle Avoidance

by

Matthew Westbrook

University of New Hampshire, September, 2020

The use of robots has become more prevalent in the last several decades in many sectors such as manufacturing, research, and consumer use [18]. With such varying environments and requirements of these robots it has become increasingly important to develop systems capable of adapting and ensuring safety of the robot and surroundings. This study examines shared control as a method of obstacle avoidance for mobile robots.

Shared control makes use of multiple control modes to obtain desired properties from each. This lends a wide range of applications of shared control, from assisted wheelchair operation [37] to autonomous vehicle navigation [10]. Shared control allows for highly versatile controllers and enables easier interfacing with humans.

In this thesis we propose control strategies for two mobile robots: a kinematic non-holonomic wheeled robot and a dynamic quad-rotor. Lyapunov analysis is used to show stability of the systems while accounting for shared control switching. With the shared control architecture, it is proven the robots always avoid collision with obstacles. The theoretical analysis is validated with experiments which show promising results and motivate shared control as a viable solution for safe navigation in other systems.

CHAPTER 1

INTRODUCTION

This chapter will provide a basic introduction to robots and navigation as the basis of this thesis. The types of robot which are the focus of this work are introduced with their basic operation. Previous navigation and obstacles avoidance methods are also discussed.

1.1 Introduction to Robots

There are many types of robots with varying capabilities in sensing, movement, and intelligence which can generally be divided into mobile and non-mobile robots. Non-mobile robots typically have a fixed base or global location. These are often robotic arms which can have multiple joints and high degrees of freedom (DOF). Mobile robots are a diverse subset of robots including many variations of wheeled, flying, or other robots. This thesis focuses on mobile robots.

There are wide applications of mobile robots in different fields. Mobile robots are commonly used for research and data collection, for example the Mars Exploration Rovers [7], and artificial satellites [2]. They are also used for industry purposes such as the Amazon Robots [1] which move shelves of inventory in a coordinated manner to improve efficiency, and consumer use like the vacuuming iRobot Roomba [6]. The focus of this thesis is on the general model of two commonly used mobile robots: the non-holonomic wheeled robot and the quad-rotor.

1.1.1 Introduction to Non-Holonomic Wheeled Robots

A holonomic robot is any robot where the controllable DOF is equal to the total DOF [17]. A non-holonomic robot is therefore a robot with less controllable DOF than total DOF. A diagram of a wheeled non-holonomic robot is shown in Fig. 1.1. The wheels rotate independently of each other in either direction on fixed axis. This robot operates in a two dimensional space lending three

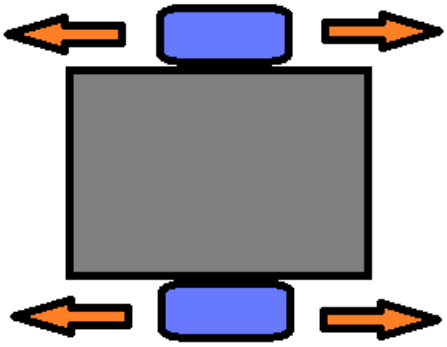

Figure 1.1: Diagram of a non-holonomic wheeled robot.

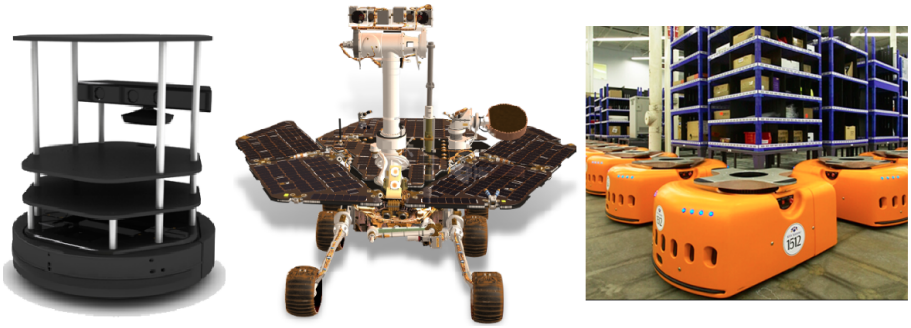

Figure 1.2: Examples of non-holonomic wheeled robots.

total degrees of freedom, two translational and one of rotation. The robot motion is constrained from moving in the direction of the wheel axis subject to assumptions about no slip discussed further in Chapter 2. As the robot can only directly control two DOF it is in the class of non-holonomic mobile robots. A third castor wheel is often used on these types of robots to prevent the robot from tipping while not imposing any constraints.

Examples of wheeled robots which may be modeled as non-holonomic are shown in Fig. 1.2. On the left is the TurtleBot 2 research robot [8], in the middle is the Mars Curiosity Rover [7], and on the right is the Amazon Robotics warehouse robot [1]. These robots all have different configurations, design, and components but share the basic structure and constraints of a non-holonomic robot. These robots may have strict requirements of maintaining obstacle avoidance and safety due to operation near people or cost of damage from collision, justifying development of obstacle avoidance techniques.

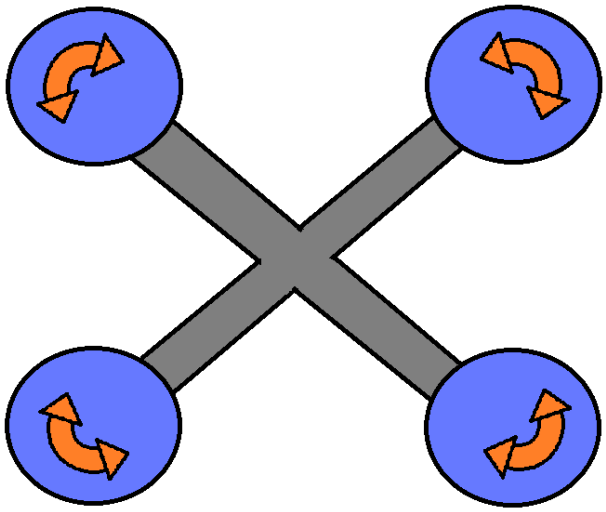

Figure 1.3: Diagram of a quad-rotor.

Figure 1.4: Examples of quad-rotors.

1.1.2 Introduction to Quad-Rotors

A quad-rotor is one type of Vertical Take-Off and Landing (VTOL) vehicle which operates in six DOF, three translational and three rotational. A diagram of a quad-rotor is shown in Fig. 1.3. Four rotors independently rotate to apply net force and moments to the quad-rotor to control it. The quad-rotor is under-actuated as there are six DOF and only four rotors. The quad-rotor is controlled by varying direction and velocity of each rotor to achieve desired net torque and force in highly coupled dynamics as discussed in Chapter 4. Holonomic or non-holonomic characterization does not apply to quad-rotors as they are free bodies subject to forces and moments and no constraints are imposed on it [17].

Examples of quad-rotors are shown in Fig. 1.4. On the left is a commercial camera quad-rotor from DJI Enterprises [4], in the middle is a small, custom built racing quad-rotor [3], and on the right is a concept emergency medical delivery quad-rotor from the Journal of Emergency

Medical Services (JEMS) [5]. The quad rotors follow a similar structure with similar components, with additions for specific applications. Quad-rotors are generally light weight and moving at high velocities, making obstacle avoidance critical.

1.2 Navigation

Navigation is a key element of mobile robotics, whether accomplished autonomously or by a human operator. At its core, navigation gets the the robot from its current state to some desired state to accomplish a task. This can be complicated by dynamics or inaccuracies in the control of the robot. It can also be complicated by problems in perception or localization. Navigation can also be affected by dynamic or unplanned obstacles in the path of the robot. The focus of navigation problems in this thesis is inaccuracies in the robot control and dynamic obstacles, ignoring aspects of perception and localization. We will discuss different known methods of navigation as well as the strengths and weaknesses of each.

1.2.1 Human Operation

The most basic method of navigation is human operation. This method involves a human directly controlling actuation of the robot. Human operation is commonly used for piloting both wheeled and flying robots.

This control method has the benefit of generally being easy to implement. Human operation also has the advantage of being highly adaptable to complex environments and objectives which are difficult to explicitly define. Complex environments may include large local minima which are difficult to escape for autonomous algorithms but straightforward to a human operator. Difficult to define problems include robot surgery [20] where an explicit objective is unclear but a human operator is capable of completing the task.

There are also many disadvantages to direct human operation. The delay for human operation versus on-board autonomous navigation may be a problem such as with the NASA Mars Exploration Rovers [7] where the delay is several minutes. With human operation there is no guarantee

on the optimality of solutions which may be obtained from autonomous control. There are also practical aspects to consider, such as cost and training of human operators.

1.2.2 Potential Fields

Potential fields or gradient descent methods are simple autonomous methods of robot navigation where the robot follows a vector field to a goal location. The basic potential field method models goals as attractive forces from every point towards the goal location and obstacles as repelling forces from every point in the opposite direction of the obstacle [43]. The sum of the forces can be taken at any point which gives the robot a vector of travel.

There are advantages of this method due to the simplicity. The gradient to follow is very fast to compute given the location of the robot as long as the goal and obstacles are well defined. This allows potential field methods to be used even with disturbance or uncertain robot motion as long as the localisation is certain. This method is also fast in implementation and can find solutions quickly in simple environments.

Potential field methods also have significant problems. As the forces are summed this method is particularly susceptible to local minima where attractive and repelling forces sum to zero. For complex environments it is very likely the robot will be directed to regions where it is not able to escape using a naive potential field method. Potential fields also make no attempt at optimality, making it difficult to determine if a solution found is a good one.

1.2.3 Trajectory Optimization

Other motion planners aim to optimize a trajectory. These planners generally start with some rough plan and modify it until an end condition is met. Trajectory optimizers work by replacing a trajectory in the plan with two trajectories which must start and end in the same states as the initial trajectory, requiring a *steering function* which, given two states, returns an optimal and feasible trajectory between them (assuming there are no obstacles). This can be done many ways such as invoking a local planner or *elastic band* methods which minimize the energy along a path. These planners can quickly find high quality solutions in many environments. An example of a trajectory

optimization algorithm is CHOMP [39] which uses co-variant gradient descent to continuously improve the path until stopped. CHOMP unlike other planners does not require the initial trajectory to be collision free and is proven locally optimal. CHOMP is state of the art shown by comparison to other trajectory optimization planners but still suffers from pitfalls of trajectory optimization as path planning.

Trajectory optimization has difficulties with complex systems and environments. As trajectory optimization requires a steering function it is unsuitable for systems with control inaccuracies or disturbance. For most systems there is no known way to connect two arbitrary states optimally, including systems with dynamics or disturbance. For systems which have a steering function there is still no guarantee that trajectory optimization will find a solution if one exists and there is no guarantee of global optimality.

1.2.4 Path Planning

Path planning provides a more complex solution to autonomous robot navigation. There are many algorithms used for path planning in different applications.

Graph based path planning [35, 31, 42, 14] finds paths in a discretized state space, often a grid. The completeness and optimality of these planners rely heavily on the discretization. These planners may plan entirely before any action [35, 31, 42], some find single optimal solutions [35], *any-time* planners search until stopped finding incrementally better sub-optimal solutions [31, 42]. Others plan in real time [14], providing an action at some predetermined interval. These methods are not suitable for robots which cannot directly follow the graph in which the solution is embedded.

This leads to sampling based planners that are not limited by a fixed discretization. Like the graph based counterparts there are some planners which do all planning before movement [27, 25, 16, 28], some aiming to quickly find a single solution [27], some continue searching to find better solutions until stopped [25]. Others, as with graph-based planners, plan in real time [34]. Some of these planners require a steering function [25, 16], while others only require the

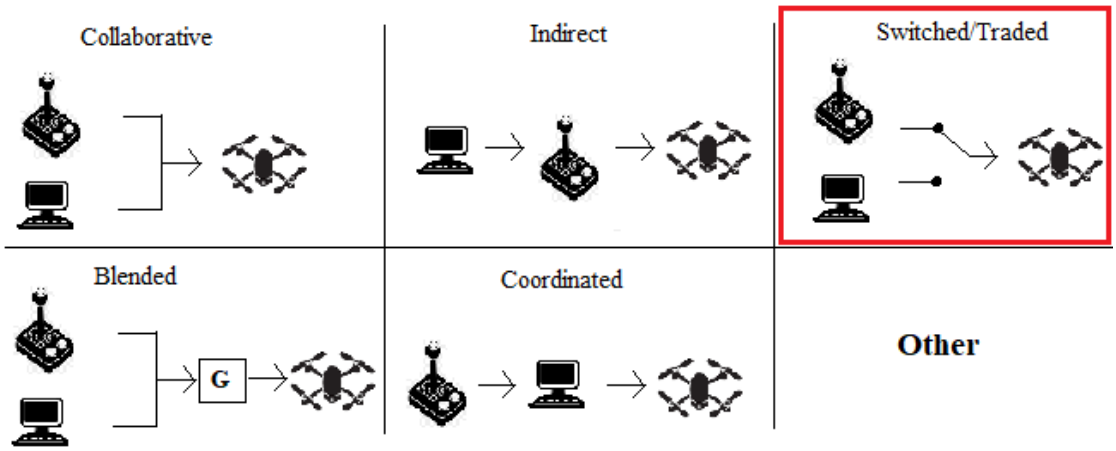

Figure 1.5: Methods of shared control.

ability to forward simulate the dynamics of the system [27, 28]. These planners at least require the ability to exactly forward simulate dynamics, making them unsuitable for robots with disturbance or uncertainty. The planners which create an entire plan before action are unsuitable for changing uncertain environments where there may be dynamic or unplanned obstacles. The real-time variants will generally still provide controls at a much slower time constant than other methods.

1.3 Introduction to Shared Control

Shared control uses multiple control methods to better adapt to the problem. Shared control often combines human operation with one or more autonomous controls in order to combine high level problem solving of human operation with the fast response time and potential optimality, safety, or hands-off advantages of autonomous systems. Shared control is effective for dynamic environments where global path planners are too slow to react [37].

There are many ways to share control methods within a system. The focus of this thesis is switching shared control outlined in red in Fig. 1.5 where the joystick represents human control while the computer represents autonomous. This control strategy gives one method full control of the robot at a time. This method is ideal for maintaining safety of a robot or surroundings as a controller can be developed just to maintain safety without also needing to reach an goal state.

There have been previous uses of shared control for safety and obstacle avoidance. A shared controller is developed to assist with safe wheelchair driving in [37]. This paper uses potential field methodology to add repulsive forces only when the wheelchair meets unsafe criteria dependent on speed and distance to obstacles due to inertia. The shared controller was implemented on a wheelchair to avoid obstacles in real time. Experiments showed the controller decreased collisions and improved the safety perception of the user [37]. This controller does not guarantee avoiding collisions and the wheelchair with the shared controller did still experience collisions in experiments. This makes this method not ideal for systems with critical requirements of avoiding collision.

Shared controllers were developed for obstacle avoidance for a non-holonomic mobile robot in [21, 23, 24]. A controller which modifies the reference signal was developed for a kinematic model of a mobile robot in [21] and for the dynamic model in [23]. These controllers where shown to avoid obstacles without *a priori* information through simulation and experiments. However, these controllers do not consider disturbance or saturation in their approach. In [24] the control switches between a safe controller and unsafe controller. The model of the robot is kinematic but includes disturbance and accounts for saturation in the control law. Simulations and experiments were conducted that showed the robot successfully avoiding obstacles and reaching objectives while switching between the safe and unsafe controller.

A shared controller is developed for the kinematic model of a quad-rotor in [22]. This paper uses a hysteresis switch in safe, hysteresis, and dangerous sets to alternate between feedback and human control. This controller is able to guarantee obstacle avoidance for the quad-rotor. Simulation showed the controller was effective for safe operation of the quad-rotor however the controller was not implemented on an actual system so it is unclear if the kinematic model is sufficient in implementation. Other control methods such as PID control [40], feedback linearization [45], or fuzzy control [13] have been implemented with quad-rotors and shown to be effective at controlling the quad-rotor to a desired state but do not consider obstacle avoidance or safety.

1.4 Contribution

The work in this thesis is motivated by the need for adaptable obstacle avoidance in robotics. Previous work has developed many methods of navigation and obstacle avoidance for varying robots and environments. Shared control has been shown to be an effective method for some systems. This work aims to broaden the scope of shared control as an obstacle avoidance method by developing it for more systems and showing how it is easily implemented with other control methods to add safety.

The goal of this work is to develop a shared controller for a kinematic wheeled robot and a dynamic quad-rotor. The controllers must be shown to be stable and must guarantee obstacle avoidance for the robot. This allows for safe navigation which can be used with other control methods. To show this, the shared controllers are implemented with multiple autonomous algorithms and human operation in simulation and experiments. The goal of the shared controllers is to maintain obstacle avoidance for each of these simulations and experiments.

1.5 Thesis Structure and Overview

The remainder of this thesis will develop a shared controller for a wheeled mobile robot and quad-rotor. Chapter 2 will go over the kinematic model of a wheeled mobile robot and the development of a shared controller for it. The controller is developed for two versions of the kinematic model, a sensor noise model and an input disturbance and input saturation model. We will then show that both controllers are stable for the switching shared control system. Chapter 3 will present simulation results of the wheeled robot shared controller. This chapter will show the shared controllers avoid obstacles in practice and that this method is adaptable with other control methods. Chapter 4 discusses multiple quad-rotor models. A translational dynamic model of the quad-rotor is used for the shared control and a higher dimensional model with rotation is used in experiments. Chapter 5 goes over the design and stability proof of a shared controller for the translational quad-rotor model. The controller is shown to be stable and avoid obstacles while accounting for input distur-

bance and input saturation. Chapter 6 presents simulation and experimental results of the shared controller. The simulations show the shared controller working for the translational model of the quad-rotor and the experiments provide evidence that this model is sufficient in practice for avoiding obstacles. Lastly Chapter 7 provides a summary of research findings and future extensions of this work.

CHAPTER 2

WHEELED ROBOT CONTROL

This chapter will discuss the kinematic model of a wheeled mobile robot. We will then go over the development and proof of stability of a shared control architecture from [24] for the kinematic model of the non-holonomic wheeled mobile robot with modifications to disturbance/noise rejection. This controller will be evaluated independently in this thesis as well as expanded upon and used to inform analysis in further controllers.

2.1 Wheeled Robot Model

The non-holonomic wheeled mobile robot model is two wheels on a fixed axis. Other wheels on actual robots are for practical purposes such as balance or weight distribution and would be redundant or not impose additional constraints in this model and are left out. The robot operates in a two dimensional work space where $p = [x, y]^T$ denotes the world position of the robot and θ denotes the orientation or the rotation from the world reference frame to the robot's reference frame. We denote the full configuration of the robot as $q = [p^T, \theta]^T$. The reference coordinate system is shown in Fig. 2.1 where ν_f is forward velocity, and ν_l is lateral velocity.

With this coordinate system the non-holonomic constraint is $\nu_l = 0$, in other words the robot cannot directly move in the lateral direction. This restriction is imposed by the drive wheels which may only rotate about the fixed drive axis. This assumes that the wheels on the robot do not slip in the lateral direction. Transformed into the world reference frame the non-holonomic constraint becomes $\dot{x} \sin\theta - \dot{y} \cos\theta = 0$.

With no lateral velocity component, the forward velocity ν_f becomes simply $\nu = \dot{x} \cos\theta + \dot{y} \sin\theta$. The rotational velocity of the robot is denoted ω. This makes the kinematic, non-holonomic equations of motion for the robot from [24],

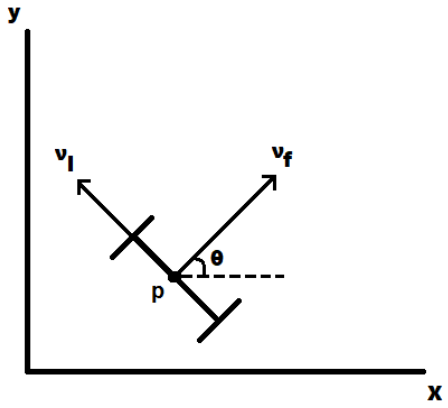

Figure 2.1: Coordinate system of wheeled mobile robot.

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \nu \\ \omega \end{bmatrix}, \tag{2.1}$$

where ν and ω are the control inputs.

2.2 Switching

The switching shared control uses two control modes, safe and unsafe. The safe mode has the objective of reaching a goal, provided autonomously or by a human. The unsafe control has the objective of avoiding collision with obstacles. The shared controller switches between safe and unsafe modes of operation depending on it's position and orientation relative to obstacles. To determine safety of the robot we define the obstacle vector,

$$\nabla\phi_o = p - p_o = \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x_o \\ y_o \end{bmatrix}, \tag{2.2}$$

where p is the robot and p_o is the closest point on the obstacle so the obstacle vector $\nabla\phi_o$ points from the robot away from the obstacle. These come from measurements from the robot and are assumed to be exact and correct. We define the distance to the obstacle, $d_o = ||\nabla\phi_o||$ where for

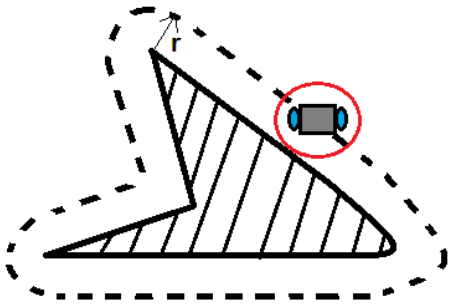

Figure 2.2: Obstacle virtual expansion.

$X = [x_1, ..., x_n]$, $||X|| = \sqrt{x_1^2 + ... + x_n^2}$, and the angle from the orientation of the robot to face the obstacle, $\theta_o = \theta - atan2(y_o - y, x_o - x)$ where,

$$atan2(y,x) = \begin{cases} arctan(\frac{y}{x}) & \text{if } x > 0, \\ arctan(\frac{y}{x}) + \pi & \text{if } x < 0 \text{ and } y \geq 0, \\ arctan(\frac{y}{x}) - \pi & \text{if } x < 0 \text{ and } y < 0, \\ +\frac{\pi}{2} & \text{if } x = 0 \text{ and } y > 0, \\ -\frac{\pi}{2} & \text{if } x = 0 \text{ and } y < 0, \\ undefined & \text{if } x = 0 \text{ and } y = 0 \end{cases} . \tag{2.3}$$

Assuming a constant radius r which is the distance the robot center must maintain from obstacles, we create an avoidance region around obstacles by virtually expanding them as shown in Fig. 2.2. This avoidance region can also be adjusted to add a buffer for sensor error in obstacle detection. We define the switching signal of the robot,

$$\zeta = \begin{cases} 0 & \text{if } |\theta_o| \geq \frac{\pi}{2} \\ & \text{or } d_o > \frac{r}{\sin(|\theta_o|)} \\ 1 & \text{else} \end{cases} , \tag{2.4}$$

which decides if the robot is safe ($\zeta = 0$) or unsafe ($\zeta = 1$). These cases state the robot is safe if the obstacle is either far away or it is greater than 90 degrees from the current direction of travel of the robot.

2.3 Tracking Vectors

For each case of the switching signal, there is a position vector that the robot tracks. In the case of safe operation, $\zeta = 0$, the robot tracks the safe vector,

$$\nabla\phi_s = p_g - p = \begin{bmatrix} x_g \\ y_g \end{bmatrix} - \begin{bmatrix} x \\ y \end{bmatrix}, \tag{2.5}$$

where p represents the robot's position, p_g is the provided goal position and the vector $\nabla\phi_s$ points from the robot to the goal.

For unsafe operation the robot tracks a position which is a projection of the safe vector away from the obstacle. For the projection we use the obstacle vector $\nabla\phi_o$ (2.2). We want to project the safe vector tangentially to the obstacle as this will direct the robot into a state where the safe condition is met while still trying to make progress to the goal. As we cannot obtain a direct inverse of the obstacle vector since it is not a square matrix, we use the left pseudo-inverse which is,

$$\nabla\phi_o^+ = (\nabla\phi_o^T\nabla\phi_o)^{-1}\nabla\phi_o^T = \frac{\nabla\phi_o^T}{||\nabla\phi_o||_2} = \frac{(p - p_o)^T}{d_o^2}. \tag{2.6}$$

The left pseudo-inverse is used to create the null space projection matrix,

$$N_L = I - \nabla\phi_o\nabla\phi_o^+ = \frac{1}{d_o^2}\begin{bmatrix} (y - y_o)^2 & -(x - x_o)(y - y_o) \\ -(x - x_o)(y - y_o) & (x - x_o)^2 \end{bmatrix}. \tag{2.7}$$

The tracking gradient in the unsafe case, $\zeta = 1$, can now be defined as the left null space projection of the safe tracking vector,

$$\nabla\phi_u = N_L\nabla\phi_s. \tag{2.8}$$

The unified tracking vector which encompasses both the safe and unsafe operation modes of the robot is defined as,

$$\Delta p = \zeta \nabla \phi_u + (1 - \zeta) \nabla \phi_s. \tag{2.9}$$

This unified tracking vector will be used for formulation of the control law.

2.4 Error Dynamics

The control law is formulated with the objective of reaching a virtual target. The target is determined by adding the unified vector, Δp (2.9), to the robot position p. This results in the desired position,

$$p_d = p + \Delta p = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \Delta p_x \\ \Delta p_y \end{bmatrix} = \begin{bmatrix} x_d \\ y_d \end{bmatrix}. \tag{2.10}$$

A desired angle, $\theta_d = atan2(\Delta p_y, \Delta p_x)$, is also created. This leads to an error signal in the robot reference frame of,

$$\epsilon = \begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \epsilon_\theta \end{bmatrix} = R(\theta) \begin{bmatrix} \Delta p \\ \theta_d - \theta \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta p_x \\ \Delta p_y \\ \theta_d - \theta \end{bmatrix} \tag{2.11}$$

Taking the derivative of the error signal we get,

$$\begin{aligned} \dot{\epsilon} &= \begin{bmatrix} -\dot{\theta}\sin(\theta)\Delta p_x + \cos(\theta)\frac{d}{dt}(\Delta p_x) + \dot{\theta}\cos(\theta)\Delta p_y + \sin(\theta)\frac{d}{dt}(\Delta p_y) \\ -\dot{\theta}\cos(\theta)\Delta p_x - \sin(\theta)\frac{d}{dt}(\Delta p_x) - \dot{\theta}\sin(\theta)\Delta p_y + \cos(\theta)\frac{d}{dt}(\Delta p_y) \\ \dot{\theta}_d - \dot{\theta} \end{bmatrix} \\ &= \begin{bmatrix} \dot{\theta}\epsilon_y + (\dot{x}_d - \dot{x})\cos(\theta) + (\dot{y}_d - \dot{y})\sin(\theta) \\ -\dot{\theta}\epsilon_x - (\dot{x}_d - \dot{x})\sin(\theta) + (\dot{y}_d - \dot{y})\cos(\theta) \\ \dot{\theta}_d - \dot{\theta} \end{bmatrix} \end{aligned} \tag{2.12}$$

We assume the virtual target trajectory follows the same equations of motion as the robot while the safety signal is not switching so that we obtain desired forward and rotational velocities satisfying,

$$\begin{bmatrix} \dot{x}_d \\ \dot{y}_d \\ \dot{\theta}_d \end{bmatrix} = \begin{bmatrix} \cos(\theta_d) & 0 \\ \sin(\theta_d) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \nu_d \\ \omega_d \end{bmatrix}. \tag{2.13}$$

By applying non-holonomic constraints, using the angle difference identity, and substituting velocities into the error dynamics (2.12) we get,

$$\begin{aligned}
\dot{\epsilon} &= \begin{bmatrix} \omega\epsilon_y - \nu + \dot{x}_d\cos(\theta) + \dot{y}_d\sin(\theta) \\ -\omega\epsilon_x - \dot{x}_d\sin(\theta) + \dot{y}_d\cos(\theta) \\ \omega_d - \omega \end{bmatrix} \\
&= \begin{bmatrix} \omega\epsilon_y - \nu + \dot{x}_d\cos(\theta_d - \epsilon_\theta) + \dot{y}_d\sin(\theta_d - \epsilon_\theta) \\ -\omega\epsilon_x - \dot{x}_d\sin(\theta_d - \epsilon_\theta) + \dot{y}_d\cos(\theta_d - \epsilon_\theta) \\ \omega_d - \omega \end{bmatrix} \\
&= \begin{bmatrix} \omega\epsilon_y - \nu + \dot{x}_d(\cos(\theta_d)\cos(\epsilon_\theta) + \sin(\theta_d)\sin(\epsilon_\theta)) + \dot{y}_d(\sin(\theta_d)\cos(\epsilon_\theta) - \cos(\theta_d)\sin(\epsilon_\theta)) \\ -\omega\epsilon_x - \dot{x}_d(\sin(\theta_d)\cos(\epsilon_\theta) - \cos(\theta_d)\sin(\epsilon_\theta)) + \dot{y}_d(\cos(\theta_d)\cos(\epsilon_\theta) + \sin(\theta_d)\sin(\epsilon_\theta)) \\ \omega_d - \omega \end{bmatrix} \\
&= \begin{bmatrix} \omega\epsilon_y - \nu + (\dot{x}_d\cos(\theta_d) + \dot{y}_d\sin(\theta_d))\cos(\epsilon_\theta) + (\dot{x}_d\sin(\theta_d) - \dot{y}_d\cos(\theta_d))\sin(\epsilon_\theta) \\ -\omega\epsilon_x + (-\dot{x}_d\sin(\theta_d) + \dot{y}_d\cos(\theta_d))\cos(\epsilon_\theta) + (\dot{x}_d\cos(\theta_d) + \dot{y}_d\sin(\theta_d))\sin(\epsilon_\theta) \\ \omega_d - \omega \end{bmatrix} \\
&= \begin{bmatrix} \dot{\epsilon}_x \\ \dot{\epsilon}_y \\ \dot{\epsilon}_\theta \end{bmatrix} = \begin{bmatrix} \omega\epsilon_y - \nu + \nu_d\cos(\epsilon_\theta) \\ -\omega\epsilon_x + \nu_d\sin(\epsilon_\theta) \\ \omega_d - \omega \end{bmatrix}
\end{aligned} \tag{2.14}$$

At this point we also noise in sensing or input disturbance by adding δ_i for $i \in \{1, 2, 3\}$ to the error signal bounded by the constant α such that $|\delta_i| \leq \alpha$. This makes the final error dynamics,

$$\dot{\epsilon} = \begin{bmatrix} \dot{\epsilon}_x \\ \dot{\epsilon}_y \\ \dot{\epsilon}_\theta \end{bmatrix} = \begin{bmatrix} \omega\epsilon_y - \nu + \nu_d \cos(\epsilon_\theta) + \delta_1 \\ -\omega\epsilon_x + \nu_d \sin(\epsilon_\theta) + \delta_2 \\ \omega_d - \omega + \delta_3 \end{bmatrix} \tag{2.15}$$

2.5 Control Laws And Stability Analysis

For stability analysis of the control system we must formulate the error as a hybrid system and find a common Lyapunov function for all subsystems [29]. For this we define the hybrid system,

$$\dot{\epsilon} = \begin{cases} f_1 & \text{if } \zeta = 0 \\ f_2 & \text{else} \end{cases}, \tag{2.16}$$

which makes subsystems f_1 and f_2 of the error derivative (2.15) for each case of the switching signal (2.4), either safe ($\zeta = 0$) or unsafe ($\zeta = 1$).

Conditions for stability under arbitrary switching are stated in [9]. In our case the switching is between safe and unsafe conditions. An open subset Ω of $\mathbb{R}^n$ is defined that contains the origin. With this subset [9] says that the Lyapunov function $V(x) : \Omega \to [0, \infty)$ is a common strict Lyapunov function for all subsystems if:

1. It is of differentiability class C^1, meaning it is differentiable everywhere
2. It is positive definite
3. $\dot{V}(x) = \Delta V(x) \cdot f_p(x) < 0$,

for all $x \in \Omega \backslash \{0\}$ and for $p \in \{1, 2\}$.

The proposed Lyapunov candidate function is,

$$V(\epsilon) = \frac{1}{2}(\epsilon_x^2 + \epsilon_y^2) + \frac{1}{k_y}(1 - \cos(\epsilon_\theta)), \tag{2.17}$$

where k_y is a positive scalar gain. It is clear that this candidate function is of differentiability class C^1 and is positive everywhere but the origin, $\epsilon = 0$ where $V(0) = 0$.

2.5.1 Control Law: Sensor Noise

This section develops a control law considering error dynamics from (2.15) with the Lyapunov candidate function (2.17) with sensor noise. The Lyapunov candidate derivative becomes,

$$\begin{aligned}\dot{V}(\epsilon) =&\epsilon_x\dot{\epsilon}_x + \epsilon_y\dot{\epsilon}_y + \frac{1}{k_y}\dot{\epsilon}_\theta \sin(\epsilon_\theta)\\ =&\epsilon_x(\omega\epsilon_y - \nu + \nu_d\cos(\epsilon_\theta) + \delta_1) + \epsilon_y(-\omega\epsilon_x + \nu_d\sin(\epsilon_\theta) + \delta_2) + ...\\ &\frac{1}{k_y}(\omega_d - \omega + \delta_3)\sin(\epsilon_\theta)\\ =&\epsilon_x(-\nu + \nu_d\cos(\epsilon_\theta) + \delta_1) + \epsilon_y(\nu_d\sin(\epsilon_\theta) + \delta_2) + \frac{\sin(\epsilon_\theta)}{k_y}(\omega_d - \omega + \delta_3).\end{aligned} \tag{2.18}$$

The control inputs are chosen as,

$$\begin{bmatrix}\nu\\ \omega\end{bmatrix} = \begin{bmatrix}k_x\epsilon_x + \nu_d\cos(\epsilon_\theta) + \frac{\alpha^2}{\lambda}\epsilon_x + \frac{\alpha^2 sgn(\epsilon_x)\epsilon_y^2}{\lambda^2}\\ \omega_d + \nu_d k_y\epsilon_y + (k_\theta + \frac{\alpha^2}{\lambda})\sin(\epsilon_\theta) + k_y\frac{\alpha^2\epsilon_y^2}{\lambda^2}sgn(\sin(\epsilon_\theta))\end{bmatrix}, \tag{2.19}$$

where k_x,k_θ, and λ are positive scalar gains with $\lambda << \alpha$. The Lyapunov candidate function derivative (2.18) can now be re-written with control inputs from (2.19),

$$\begin{aligned}\dot{V}(\epsilon) =&\epsilon_x(-k_x\epsilon_x - \frac{\alpha^2}{\lambda}\epsilon_x - \frac{\alpha^2 sgn(\epsilon_x)\epsilon_y^2}{\lambda^2} + \delta_1) + \epsilon_y(\nu_d\sin(\epsilon_\theta) + \delta_2) + ...\\ &\frac{\sin(\epsilon_\theta)}{k_y}(-\nu_d k_y\epsilon_y - (k_\theta + \frac{\alpha^2}{\lambda})\sin(\epsilon_\theta) - k_y\frac{\alpha^2\epsilon_y^2}{\lambda^2}sgn(\sin(\epsilon_\theta)) + \delta_3)\\ =&-k_x\epsilon_x^2 - \frac{k_\theta}{k_y}\sin^2(\epsilon_\theta) + \epsilon_x\delta_1 - \frac{\alpha^2}{\lambda}\epsilon_x^2 + \epsilon_y\delta_2 - \frac{\alpha^2}{\lambda k_y}\sin^2(\epsilon_\theta) - ...\\ &\frac{\alpha^2(\epsilon_x sgn(\epsilon_x) + \sin(\epsilon_\theta)sgn(\sin(\epsilon_\theta)))}{\lambda^2}\epsilon_y^2 + \frac{\sin(\epsilon_\theta)\delta_3}{k_y}.\end{aligned} \tag{2.20}$$

This Lyapunov candidate function can be upper bounded by substituting noise δ_i for $i \in \{1, 2, 3\}$ with α resulting in,

$$
\begin{aligned}
\dot{V}(\epsilon) \leq & -k_x\epsilon_x^2 - \frac{k_\theta}{k_y}\sin^2(\epsilon_\theta) + |\epsilon_x|\alpha - \frac{\alpha^2}{\lambda}\epsilon_x^2 + |\epsilon_y|\alpha - \frac{\alpha^2}{\lambda k_y}\sin^2(\epsilon_\theta) - ... \\
& \frac{\alpha^2(|\epsilon_x| + |\sin(\epsilon_\theta)|)}{\lambda^2}\epsilon_y^2 + \frac{|\sin(\epsilon_\theta)|\alpha}{k_y} \\
\leq & -k_x\epsilon_x^2 - \frac{k_\theta}{k_y}\sin^2(\epsilon_\theta) + |\epsilon_x|\alpha(1 - \frac{\alpha}{\lambda}|\epsilon_x|) + ... \\
& |\epsilon_y|\alpha(1 - \frac{\alpha(|\epsilon_x| + |\sin(\epsilon_\theta)|)}{\lambda^2}|\epsilon_y|) + ... \\
& \frac{|\sin(\epsilon_\theta)|\alpha}{k_y}(1 - \frac{\alpha}{\lambda}|\sin(\epsilon_\theta)|).
\end{aligned} \tag{2.21}
$$

From this result we can see that if the conditions $|\epsilon_x| > \frac{\lambda}{\alpha}$, $|\sin(\epsilon_\theta)| > \frac{\lambda}{\alpha}$, and $|\epsilon_y| > \frac{\lambda^2}{\alpha(|\epsilon_x|+|\sin(\epsilon_\theta)|)}$ are met than $\dot{V}(\epsilon) < 0 \, \forall \epsilon \in \mathbb{R}^2 \times \mathbb{S}$. This leads to a maximum value $max(|\epsilon_x|\alpha(1-\frac{\alpha}{\lambda}|\epsilon_x|), |\epsilon_y|\alpha(1-\frac{2}{\lambda}|\epsilon_y|), \frac{|\sin(\epsilon_\theta)|\alpha}{k_y}(1-\frac{\alpha}{\lambda}|\sin(\epsilon_\theta)|) < \eta$ where η upper bounds all three terms and is tunable depending on chosen gain values. The final Lyapunov candidate function derivative can be upper bounded from (2.21) as,

$$
\dot{V}(\epsilon) \leq -k_x\epsilon_x^2 - \frac{k_\theta}{k_y}\sin^2(\epsilon_\theta) + 3\eta, \tag{2.22}
$$

proving $\dot{V}(\epsilon) < 0$ outside of some ball. This shows that the system with sensor noise is Globally Uniformly Ultimately Bounded (GUUB).

2.5.2 Control Law: Input Disturbance and Input Saturation

This section develops a control law considering error dynamics from (2.15) with the Lyapunov candidate function (2.17) with input disturbance and also considering input saturation. As the disturbances are from the input, $\delta_2 = 0$ because the input is only forward velocity and rotational velocity and has no lateral component. The input references ν_d and ω_d are saturated to $\nu_{max}\sigma(\nu_d)$ and $\omega_{max}\sigma(\omega_d)$ respectively where $\sigma(x)$ is a smooth saturation function meeting the criteria from [24],

$$\sigma(x): \begin{cases} \sigma(x) = 0 & \text{if } x = 0 \\ \sigma(x)x > 0 & \text{if } x \neq 0 \\ \sigma(-x) = -\sigma(x) & \\ (x-y)(\sigma(x) - \sigma(y)) \geq 0 & \cdot \\ |\sigma(x)| \leq 1 & \\ |\frac{\sigma(x)}{x}| \leq 1 & \\ |\frac{d\sigma(x)}{dx}| \leq 1 & \end{cases} \tag{2.23}$$

The function $\sigma(x) = \frac{x}{\sqrt{x^2+1}}$ is used in this paper. This makes the Lyapunov candidate derivative,

$$\begin{aligned} \dot{V}(\epsilon) =& \epsilon_x \dot{\epsilon}_x + \epsilon_y \dot{\epsilon}_y + \frac{1}{k_y} \dot{\epsilon}_\theta \sin(\epsilon_\theta) \\ =& \epsilon_x(\omega \epsilon_y - \nu + \nu_{max}\sigma(\nu_d)\cos(\epsilon_\theta) + \delta_1) + \epsilon_y(-\omega\epsilon_x + \nu_{max}\sigma(\nu_d)\sin(\epsilon_\theta)) + ... \\ & \frac{1}{k_y}(\omega_{max}\sigma(\omega_d) - \omega + \delta_3)\sin(\epsilon_\theta) \\ =& \epsilon_x(-\nu + \nu_{max}\sigma(\nu_d)\cos(\epsilon_\theta) + \delta_1) + \epsilon_y(\nu_{max}\sigma(\nu_d)\sin(\epsilon_\theta) + \delta_2) + ... \\ & \frac{\sin(\epsilon_\theta)}{k_y}(\omega_{max}\sigma(\omega_d) - \omega + \delta_3). \end{aligned} \tag{2.24}$$

The control inputs are chosen as,

$$\begin{bmatrix} \nu \\ \omega \end{bmatrix} = \begin{bmatrix} k_x\sigma(\epsilon_x) + \nu_{max}\sigma(\nu_d)\cos(\epsilon_\theta) + \frac{\alpha^2|\epsilon_x|}{\alpha|\epsilon_x|+c_1} \\ \omega_{max}\sigma(\omega_d) + \nu_{max}\sigma(\nu_d)k_y\epsilon_y + k_\theta\sin(\epsilon_\theta) + \frac{\alpha^2|\sin(\epsilon_\theta)|}{\alpha|\sin(\epsilon_\theta)|+c_2} \end{bmatrix}, \tag{2.25}$$

where k_x and k_θ are positive scalar gains and c_1 and c_2 are small positive scalars. The Lyapunov candidate function derivative (2.24) can now be re-written with control inputs from (2.25),

$$
\begin{aligned}
\dot{V}(\epsilon) =& \epsilon_x(-k_x\sigma(\epsilon_x) - \frac{\alpha^2|\epsilon_x|}{\alpha|\epsilon_x| + c_1} + \delta_1) + ... \\
& \frac{1}{k_y}(-k_\theta \sin(\epsilon_\theta) - \frac{\alpha^2|\sin(\epsilon_\theta)|}{\alpha|\sin(\epsilon_\theta)| + c_2} + \delta_3)\sin(\epsilon_\theta) \\
=& - k_x\epsilon_x\sigma(\epsilon_x) + \epsilon_x(\delta_1 - \frac{\alpha^2|\epsilon_x|}{\alpha|\epsilon_x| + c_1}) - ... \\
& \frac{k_\theta}{k_y}\sin^2(\epsilon_\theta) + \frac{\sin(\epsilon_\theta)}{k_y}(\delta_3 - \frac{\alpha^2|\sin(\epsilon_\theta)|}{\alpha|\sin(\epsilon_\theta)| + c_2})
\end{aligned}
\tag{2.26}
$$

This Lyapunov candidate function can be upper bounded by substituting disturbances δ_i with α for $i \in \{1, 2, 3\}$ resulting in,

$$
\begin{aligned}
\dot{V}(\epsilon) \leq& - k_x\epsilon_x\sigma(\epsilon_x) + |\epsilon_x|(\alpha - \frac{\alpha^2|\epsilon_x|}{\alpha|\epsilon_x| + c_1}) - ... \\
& \frac{k_\theta}{k_y}\sin^2(\epsilon_\theta) + \frac{|\sin(\epsilon_\theta)|}{k_y}(\alpha - \frac{\alpha^2|\sin(\epsilon_\theta)|}{\alpha|\sin(\epsilon_\theta)| + c_2}) \\
\leq& - k_x\epsilon_x\sigma(\epsilon_x) + \alpha|\epsilon_x|(1 - \frac{\alpha|\epsilon_x|}{\alpha|\epsilon_x| + c_1}) - ... \\
& \frac{k_\theta}{k_y}\sin^2(\epsilon_\theta) + \frac{\alpha|\sin(\epsilon_\theta)|}{k_y}(1 - \frac{\alpha|\sin(\epsilon_\theta)|}{\alpha|\sin(\epsilon_\theta)| + c_2}).
\end{aligned}
\tag{2.27}
$$

From this we know that $\alpha|\epsilon_x|(1 - \frac{\alpha|\epsilon_x|}{\alpha|\epsilon_x|+c_1}) \leq c_1$ and $\frac{\alpha|\sin(\epsilon_\theta)|}{k_y}(1 - \frac{\alpha|\sin(\epsilon_\theta)|}{\alpha|\sin(\epsilon_\theta)|+c_2}) \leq \frac{c_2}{k_y}$. This can be used to upper bound the Lyapunov candidate function derivative from (2.27) as,

$$
\dot{V}(\epsilon) \leq -k_x\epsilon_x\sigma(\epsilon_x) - \frac{k_\theta}{k_y}\sin^2(\epsilon_\theta) + c_1 + \frac{c_2}{k_y}, \tag{2.28}
$$

proving $\dot{V}(\epsilon) < 0$ outside of some ball. This shows that the system with sensor noise is also GUUB.

CHAPTER 3

WHEELED ROBOT EXPERIMENTS

This chapter examines simulation results of the wheeled mobile robot with both the sensor noise controller and the controller with input disturbance and saturation. The simulations will show how the controller works to ensure obstacle avoidance for the robot while navigating to a desired position. The simulations will also compare the two control models.

3.1 Simulation

Simulations were conducted with Matlab. For the sensor noise model the forward and rotational velocities were cut off at $|\nu| \leq \nu_{max}$ and $|\omega| \leq \omega_{max}$. This is to simulate a real system although it is not included in the formal analysis. This allows us to determine if the model is sufficient in practice.

The first simulation compares the wheeled robot sensor noise control to the input disturbance and saturation control. The sensor noise is simulated as random with bounds $-0.03 \leq \delta_{1,2,3} \leq 0.03$ and disturbance is random with $-0.03 \leq \delta_{1,3} \leq 0.03$ and $\delta_2 = 0$. To bound these disturbances and noise we choose $\alpha = 0.3$ for both controllers. The control gains are k_x, k_y, k_θ are set as $0.3, 0.4, 0.5$ respectively for both controllers. For the sensor noise control we set $\lambda = 0.03$ to be an order of magnitude less than α. For the input disturbance and saturation control we set the constants $c_1, c_2 = 0.01$. The maximum forward and angular velocities are $\nu_{max} = 0.8\frac{m}{s}$ and $\omega_{max} = \frac{\pi}{2}\frac{rad}{s}$ for a cut-off in the sensor noise control and as the desired velocities saturation values for the input saturation control. The simulation is run for the robot reaching a static goal at $[x_g, y_g] = [10, 10](m)$ from a starting position of $[x, y] = [0, 0](m)$ with an initial angle $\theta = 0$. There is an obstacle in the path with an avoidance region of $r = 0.5m$, which will trigger an unsafe condition for both controllers during traversal.

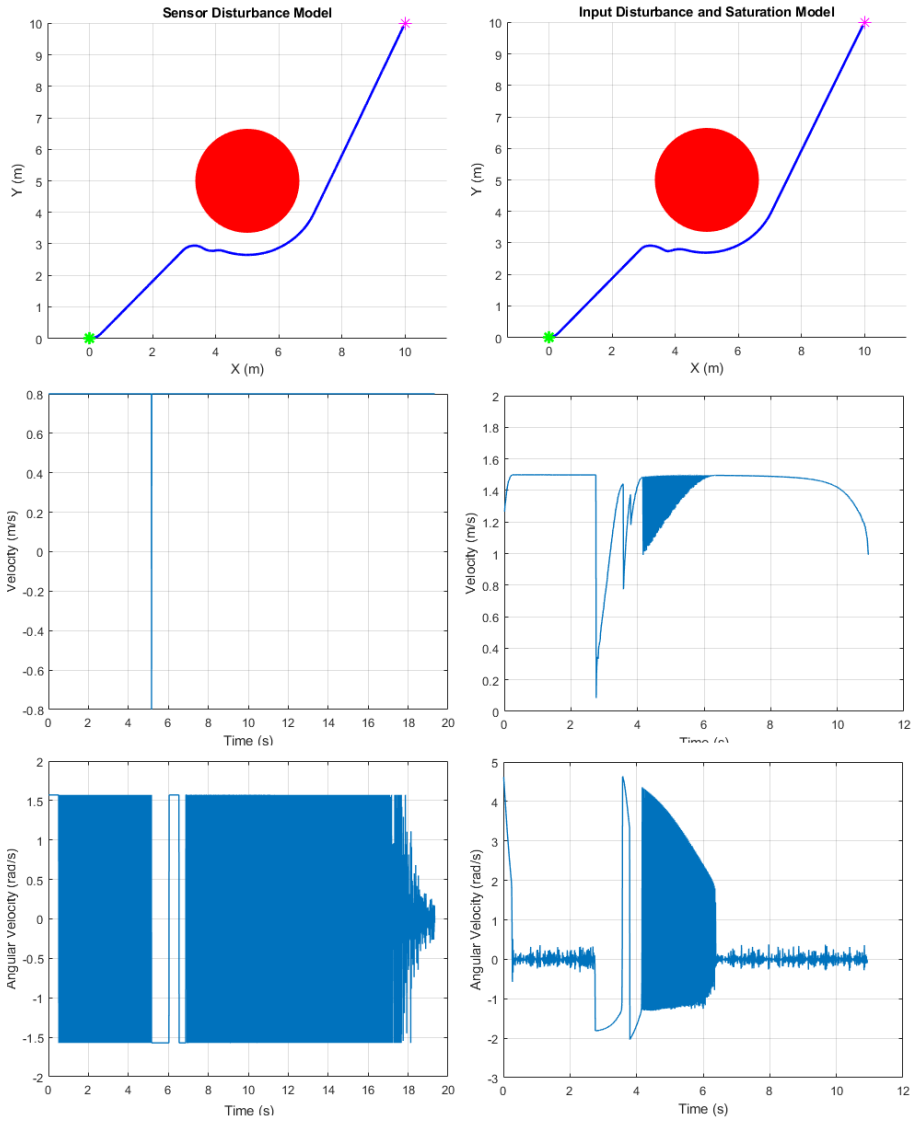

Figure 3.1: Comparison of wheeled robot controllers.

The results of the controller comparison simulation are shown in Fig. 3.1. The left column shows the sensor noise controller and the right column shows the input disturbance and saturation controller. The simulated robot path is shown as the blue line with the initial position as a green asterisk and the final goal position as a magenta asterisk. The obstacles are shown as red circles. The velocity inputs, both forward and angular, are shown for both controllers. The figure shows that both controllers result in very similar paths around the obstacle to the goal. However, the velocity inputs are different with each controller. The sensor noise controller is not designed with maximum velocities in mind and is therefore almost always providing velocities at the cutoff value. The sensing noise also causes the angular velocity to rapidly switch between positive and negative disturbance. The input disturbance and saturation controller is more consistently at low rotational velocity with the input disturbance showing. The input disturbance and saturation controller shows rapid changes in the velocity input while the robot is avoiding the obstacle. For both controllers the robot successfully avoids the obstacle and reaches the goal location. The forward velocity plot shows that the input disturbance and saturation controller has a higher velocity than the sensor noise controller. This is because the desired velocity is saturated, which can result in higher input velocities where the sensor noise controller cuts off the input velocity. This results in the input disturbance and saturation controller reaching the goal faster than the sensor noise controller, shown by the time at the bottom of the plots.

The next simulation is the wheeled robot traversing through an obstacle field to a static goal location. The robot starts in an initial position of $[x, y] = [0, 0](m)$ with an initial angle $\theta = 0$ and a goal location of $[x_g, y_g] = [30, -3](m)$. The input disturbance and saturation model of the robot is used with random disturbance $-0.03 \leq \delta_{1,3} \leq 0.03$. To bound these disturbances we choose $\alpha = 0.3$. The control gains are k_x, k_y, k_θ are set as $0.3, 0.4, 0.5$ respectively. The constants c_1 and c_2 are set to 0.01. The maximum forward and angular velocities are $\nu_{max} = 0.8\frac{m}{s}$ and $\omega_{max} = \frac{\pi}{2}\frac{rad}{s}$. The avoidance region is set to $r = 0.35m$.

The results of the obstacle field simulation are shown in Fig. 3.2. The simulated robot path is shown as the blue line with the initial position as a green asterisk and the final goal position as a

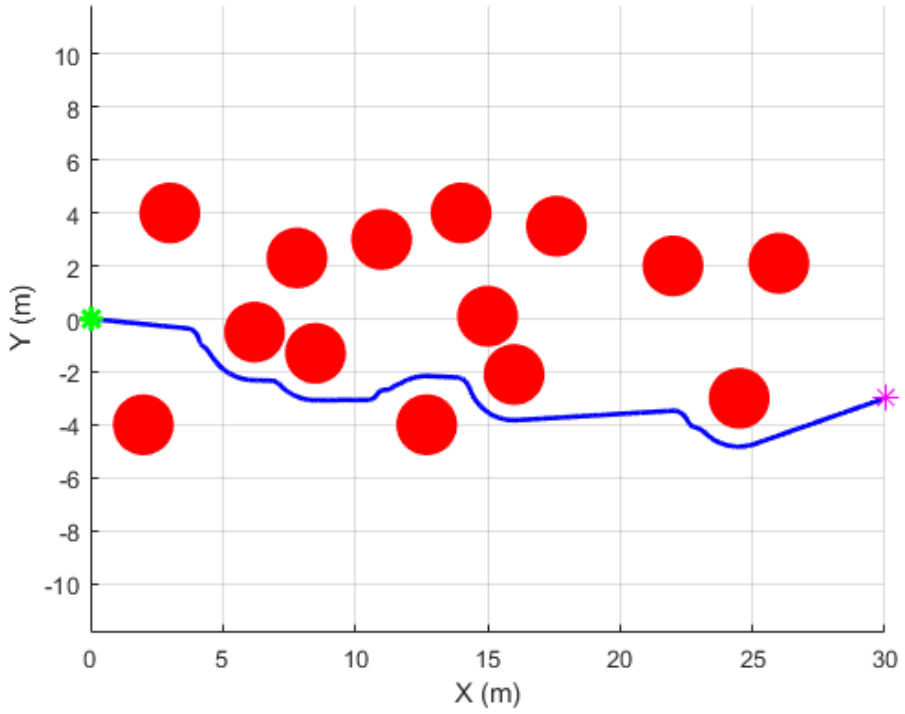

Figure 3.2: Simulation of wheeled robot in obstacle field: path.

magenta asterisk. The obstacles are shown as red circles. The robot begins on a direct path to the goal and switches to obstacle avoidance at several points along traversal. The obstacle avoidance control is not overly conservative in that it stays just outside the avoidance region of obstacles so the path does not become unnecessarily sub-optimal. The null space projection of the safe vector in the unsafe case results in the robot avoiding obstacles in an efficient manor and the robot is able to reach the goal eventually.

The next experiment uses tele-operation as the safe control for the wheeled robot. This is intended to show the adaptability of shared control to easily mix with other control methods. This allows the versatility of human operation while still guaranteeing obstacle avoidance. The safe gradient was provided as a constant magnitude of $1m$ in the direction given by a keyboard input in real time. The commands where either up $(+y)$, down $(-y)$, left $(-x)$, or right$(+x)$. The parameters are the same as the obstacle field simulation. The starting position of the robot is $[x, y] = [0, 0](m)$ with an initial angle of $\theta = 0$.

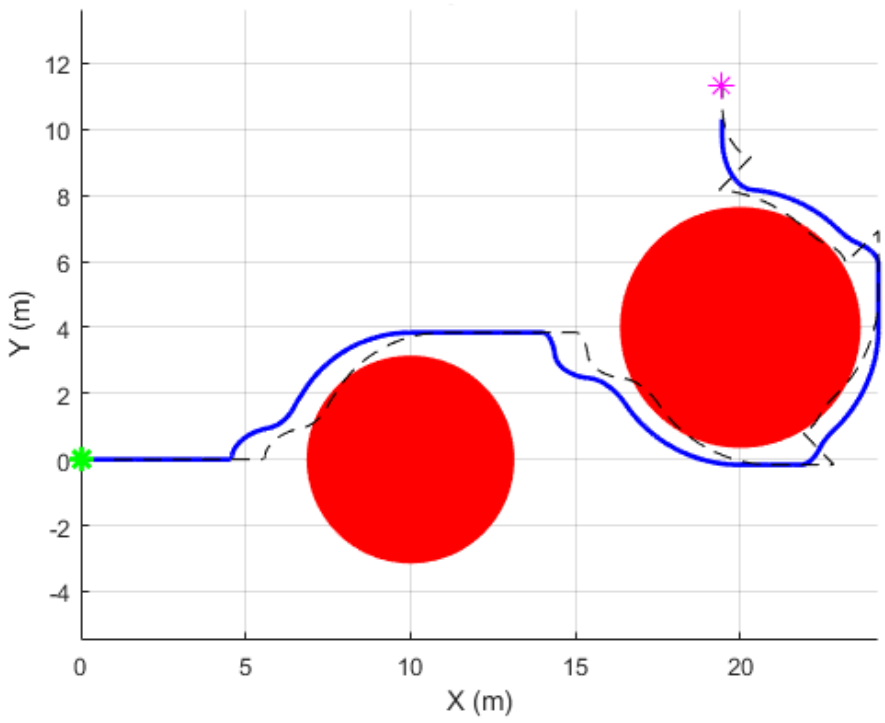

Figure 3.3: Simulation of wheeled robot tele-operation: path.

The results of the tele-operation simulation are shown in Fig. 3.3. The tele-operation reference is shown as the dotted black line and the simulated robot path is shown in blue with the initial position as a green asterisk and the final position as a magenta asterisk. The reference always stays $1m$ ahead of the robot as it is directly provided. The jumps in the reference line are when the tele-operation commands change. The shared control easily adapts to changing desired positions while ensuring obstacles are avoided.

The path components of the tele-operation simulation are shown in Fig. 3.4 where the reference is shown as the orange dotted line and the solid blue line shows the simulated robot path. The reference is always $1m$ away from the robot in either x or y. The shared control brings the robot towards the desired vector but always ensures safety.

For the next simulation the shared control is used in conjunction with path planning. This will show how the shared control can easily be implemented with other control methods to ensure obstacle avoidance and safety. A path planner is used to find a solution to a query of an initial

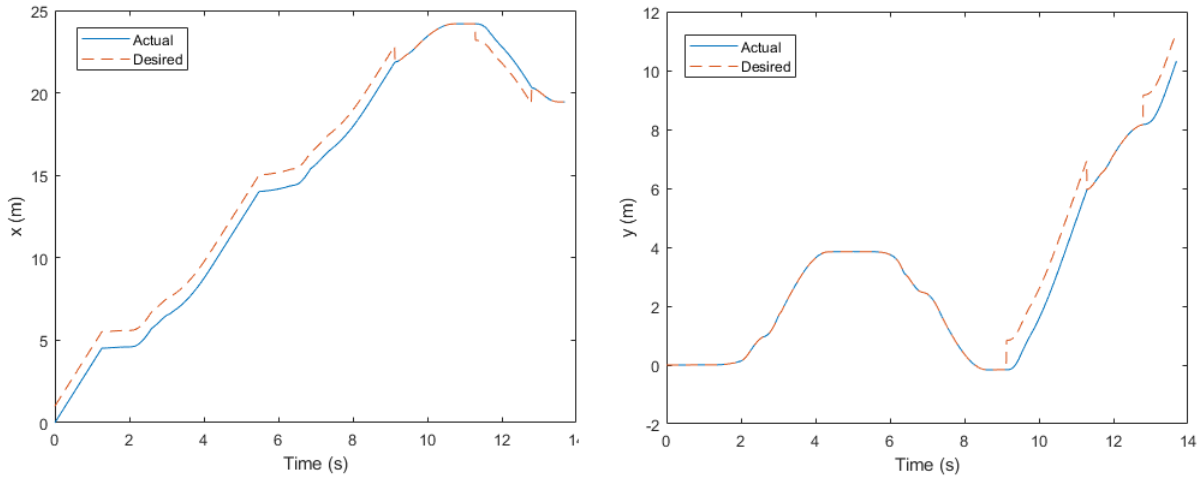

Figure 3.4: Simulation of wheeled robot tele-operation: path components.

Algorithm 1 RRT $(x_{start}, x_{goal}, r_{goal}, \mathbb{M})$

$\tau \leftarrow \{x_{start}\}$
$\pi_{sol} \leftarrow \emptyset$
while $\pi_{sol} = \emptyset$ **do**
  $x_s \leftarrow Sample(\mathbb{M})$
  $x_{near} \leftarrow Nearest(\tau, x_s)$
  if $CollisionFree(\pi(x_{near}, x_s), \mathbb{M})$ **then**
    $x_s.parent \leftarrow x_{near}$
    $x_s.\pi \leftarrow x_{near}.\pi \cup \pi(x_{near}, x_s)$
    $\tau \leftarrow \tau \cup x_s$
    if $Distance(x_s, x_{goal}) \leq r_{goal}$ **then**
      $\pi_{sol} \leftarrow x_s.\pi$
return π_{sol}

state x_{start}, a goal state x_{goal}, a goal radius r_{goal}, and a map $\mathbb{M}$. The path planner used is an implementation of RRT [27] in Matlab shown in Algorithm 1. The algorithm starts by initializing the motion tree, τ, to the start state and the solution, π_{sol} to empty. The algorithm then searches until a solution is found. The search consists of sampling a state, x_s, randomly within the bounds of the map. Euclidean distance of the xy positions is then used to determine the nearest state, x_{near}. If a straight path between the points is collision free then the sampled state is added to the motion tree. If the new state is within the goal radius of the goal state, a solution has been found. RRT is proven probabilistically complete in most cases [44], including this simulation, so it will find a

solution if one exists. It is intended to find a single solution and is therefore not guaranteed to find an optimal solution.

The map provided to the planning algorithm may not be completely accurate. The shared control guarantees that the robot will not collide with dynamic or unplanned obstacles. This is useful since when the robot encounters dynamics obstacles, it may be inefficient to take the time to find a new plan, or may cause collision if the robot cannot stop and wait for the planning algorithm. The motion plan creates way points which become the goal position for the safe operation of the shared control to track.

The experiment is done in the common motion planning environment, Bug Trap, with additional dynamic obstacles. This environment is a common example of large local minima which make finding solutions difficult for naive planners such a potential field methods. The model and parameters are the same as the obstacle field experiment. The starting position of the robot is $[x, y] = [60, 50](m)$ and the goal position is $[x, y] = [40, 50](m)$. The RRT planning algorithm is run until a solution is found. A goal bias is implemented so that five percent of the sampled states are sampled as the goal state. This is intended to speed up the search and ensure samples at the goal. Once a solution is found it is sequentially given as the goal position of the shared controller.

The results of the Bug Trap simulation are shown in Fig. 3.5. The RRT motion tree is shown in light grey and the simulated robot path is shown in blue with the start position as the green asterisk and the ultimate goal position as a magenta asterisk. The map obstacles are shown as black and the dynamic or unplanned obstacles are shown as red. The RRT plan finds a path to the goal around the large local minima. Once the robot begins traversal of the RRT generated path it senses additional obstacles which were unplanned. The unsafe mode of the shared control takes over and successfully diverts the robot around the obstacles. Obstacles are avoided throughout traversing the way points and the robot eventually reaches the goal location.

Simply providing the final goal location to the shared controller would be insufficient for reaching the goal. The shared controller would maintain safety of the robot but it would not be able to escape the local minima. The shared control can not make the decision to greatly increase the error

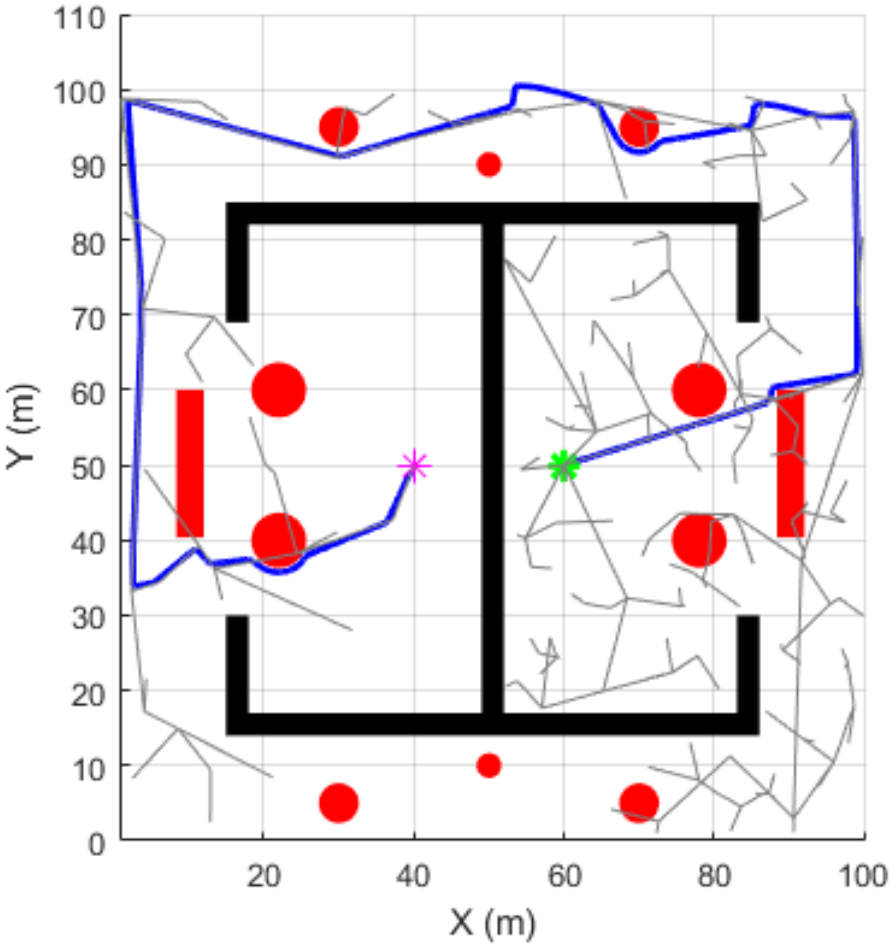

Figure 3.5: Simulation of wheeled robot in Bug Trap environment: path.

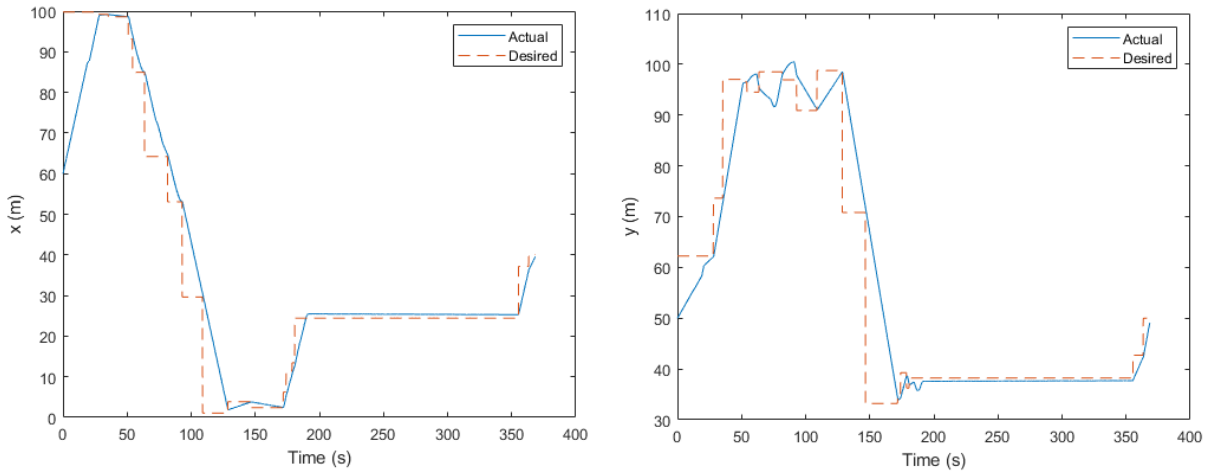

Figure 3.6: Simulation of wheeled robot in Bug Trap environment: path components.

to search for a different way to the goal. RRT alone does not provide a way to adapt to dynamic obstacles and would cause collision during traversal. This shows the usefulness of shared control for combining controllers.

The path components of the Bug Trap experiment are shown in Fig. 3.6. The orange dotted line shows the reference as the robot reaches each way point and the blue solid line is the simulated robot path. When possible the robot follows directly to each way point and when dynamic obstacles are encountered, the robot finds an efficient path around them.

CHAPTER 4

QUAD-ROTOR MODEL

This chapter will develop the models used for the quad-rotor. There are two models used in this thesis, a simplified translational model used in the shared controller, and a higher dimensional rotational model which is used in implementation. The translational model enables the shared controller to avoid obstacles while accounting for inertia while the rotational models allows for stable control of an actual quad-rotor.

4.1 Translational Model

The shared control considers a translational model of the quad-rotor with dynamics. The dynamics are not concerned with the rotation of the quad-rotor as it is assumed there is a constant radius the center of the robot must maintain from obstacles to avoid collision. The control is decoupled in each direction therefore $i \in (x, y, z)$ is used to denote the i component of the control for analysis. We consider the simplified double integrator model,

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} u_x - \delta_x \\ u_y - \delta_y \\ u_z - \delta_z \end{bmatrix}, \tag{4.1}$$

where u_i and δ_i stand for the control input acceleration and bounded input disturbance, respectively. The control u_i has a saturation value $|u_i| \leq u_{i_{max}}$. Disturbance is bounded by $|\delta_i| \leq \alpha_i$. Unlike the wheeled robot, it is important to consider dynamics with acceleration due to the nature of quad-rotors and aircraft in general. As they are fast moving and have non-negligible times to decelerate, neglecting to consider acceleration in the model would likely lead to failure in implementation.

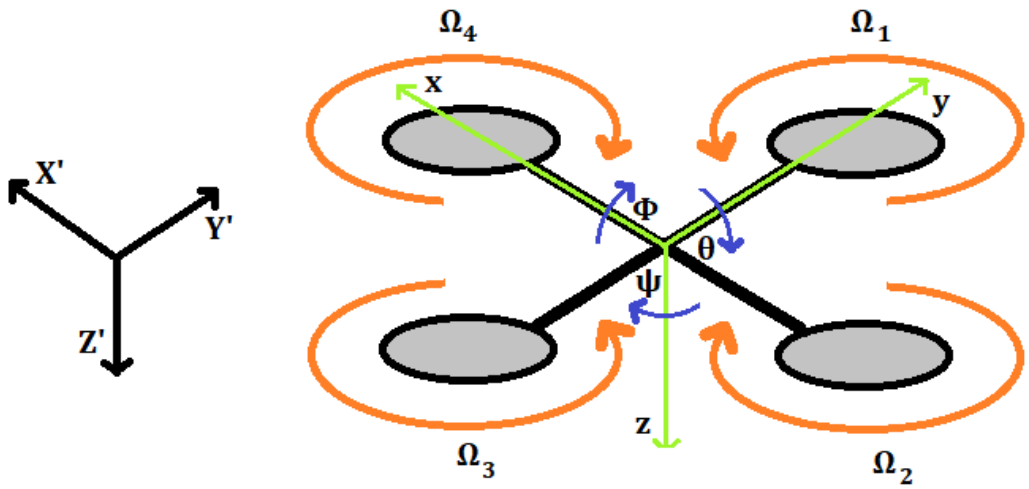

Figure 4.1: Rotational coordinate system of quad-rotor.

4.2 Rotational Model

For implementation the shared control operates as the inner loop of a controller which handles the more complex dynamics of the quad-rotor. The rotational coordinate system for this model from [38] is shown in Fig. 4.1 where the global coordinates are denoted as $[X', Y', Z']$ and robot coordinates are denoted as $[x, y, z]$. For this model we assume The robot operates at small ϕ (pitch) and θ (roll).The orientation of the quad-rotor is defined in reference to global coordinates by a 1-2-3 rotation from the Euler angles ϕ (pitch), θ (roll), then ψ (yaw) in that order. This creates the rotation matrix,

$$R = \begin{bmatrix} \cos(\theta)\sin(\psi) & \sin(\phi)\sin(\theta)\cos(\psi) & \cos(\phi)\sin(\theta)\cos(\psi) + \sin(\phi)\sin(\psi) \\ \cos(\theta)\sin(\psi) & \sin(\phi)\sin(\theta)\sin(\psi) + \cos(\theta)\cos(\psi) & \cos(\phi)\sin(\theta)\sin(\psi) - \sin(\phi)\cos(\psi) \\ -\sin(\theta) & \sin(\phi)\cos(\theta) & \cos(\phi)\cos(\theta) \end{bmatrix} \tag{4.2}$$

As the robot operates at small pitch and roll angles we can assume that body rates are equal to the Euler angle rates.

We now define the force, F, and moment, M, of each rotor as,

$$
\begin{aligned}
F =& \frac{1}{2}\rho A C_T r_p^2 \Omega_k^2 = K_F \Omega_i^2 \\
M =& \frac{1}{2}\rho A C_D r_p^2 \Omega_k^2 = K_M \Omega_i^2,
\end{aligned}
\tag{4.3}
$$

for $k \in [1, 2, 3, 4]$ where ρ is the air density, A is the surface area of the propeller blade, C_T is the thrust coefficient, C_D is the drag coefficient, r_p is the radius of the propeller, and Ω_k is the velocity of rotor k. The constants K_F and K_M assume constant coefficients and combine them to only leave two force and moment constants which may be calibrated. Net forces are exerted on the quad-rotor by creating different forces and moments on different rotors. The only net force which can be applied is in the z direction resulting in the net body forces,

$$
F_b = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ K_F(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -U_1 \end{bmatrix}. \tag{4.4}
$$

The net body moments are,

$$
M_b = \begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} K_F l(\Omega_3^2 - \Omega_1^2) \\ K_F l(\Omega_4^2 - \Omega_2^2) \\ K_M(\Omega_1^2 - \Omega_3^2 + \Omega_3^2 - \Omega_4^2) \end{bmatrix} = \begin{bmatrix} lU_2 \\ lU_3 \\ U_4 \end{bmatrix}, \tag{4.5}
$$

where l is the length of the quad-rotor arm from the origin of the body to the center of the rotor. These net forces and torques include the system input vector,

$$
u = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix}, \tag{4.6}
$$

The dynamics are given as,

$$\begin{aligned} M_b =& J\dot{\omega} + \omega X J\omega \\ m\ddot{r} =& mg + RF_b, \end{aligned} \tag{4.7}$$

where J is the quad-rotor's inertia matrix, m is the mass of the quad-rotor, and g is the gravitational constant. The inertia matrix is defined,

$$J = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}, \tag{4.8}$$

where I_{xx}, I_{yy}, and I_{zz} are the principal moments of inertia for the x, y, and z axis respectively. Inserting the inertia matrix (4.8) into the rotational dynamics in (4.7) results in,

$$\begin{bmatrix} lU_2 \\ lU_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} + \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} X \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}. \tag{4.9}$$

Solving this system of equations for the rotational accelerations gets,

$$\begin{aligned} \ddot{\phi} =& \frac{l}{I_{xx}} U_2 + \frac{I_{yy} - I_{zz}}{I_{xx}} \dot{\theta}\dot{\psi} \\ \ddot{\theta} =& \frac{l}{I_{yy}} U_3 + \frac{I_{zz} - I_{xx}}{I_{yy}} \dot{\phi}\dot{\psi} \\ \ddot{\psi} =& \frac{1}{I_{zz}} U_4 + \frac{I_{xx} - I_{yy}}{I_{zz}} \dot{\theta}\dot{\psi}. \end{aligned} \tag{4.10}$$

With $\cos(x)$ abbreviated as $c(x)$ and $\sin(x)$ abbreviated as $s(x)$ and,

$$R = \begin{bmatrix} c(\theta)s(\psi) & s(\phi)s(\theta)c(\psi) & c(\phi)s(\theta)c(\psi) + s(\phi)s(\psi) \\ c(\theta)s(\psi) & s(\phi)s(\theta)s(\psi) + c(\theta)c(\psi) & c(\phi)s(\theta)s(\psi) - s(\phi)c(\psi) \\ -s(\theta) & s(\phi)c(\theta) & c(\phi)c(\theta) \end{bmatrix}, \tag{4.11}$$

we can write out the translational dynamics as,

$$\begin{bmatrix} m\ddot{x} \\ m\ddot{y} \\ m\ddot{z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + R \begin{bmatrix} 0 \\ 0 \\ -U_1 \end{bmatrix} \quad , \tag{4.12}$$

which solving for the translational accelerations becomes,

$$\begin{aligned} \ddot{x} &= -\frac{U_1}{m}(\sin(\phi)\sin(\psi) + \cos(\phi)\cos(\psi)\sin(\theta)) \\ \ddot{y} &= -\frac{U_1}{m}(\cos(\phi)\sin(\theta)\sin(\psi) - \cos(\psi)\sin(\phi)) \\ \ddot{z} &= -\frac{U_1}{m}(\cos(\phi)\cos(\theta)) + g. \end{aligned} \tag{4.13}$$

CHAPTER 5

QUAD-ROTOR CONTROL

This chapter will develop and prove stability of a shared control architecture for a dynamic model of a quad-rotor. The controller will follow a similar analysis to the wheeled mobile robot while accounting for double integrator dynamics.

5.1 Switching

The shared controller switches between unsafe and safe modes of operation depending on it's position and velocity, both magnitude and direction, relative to obstacles. To determine safety of the robot we must define an unsafe vector, and since the model includes inertia this requires taking the velocity of the robot into account. We define $\nu = [\dot{x}, \dot{y}, \dot{z}]^T$ as the velocity vector of the robot. We create an avoidance region around obstacles by virtually expanding them a constant radius, r, in all directions to account for the size of the robot. The set $\mathbb{P}$ is all points on the obstacle avoidance region and for each $p_k \in \mathbb{P}$, $\nabla\phi_k = p - p_k$ represents the vector from the robot to the point p_k. The maximum avoidance acceleration, $U_{k_{max}}$, of the robot in the direction $-\nabla\phi_k$ is found by the intersection of $-c\nabla\phi_k$ and the control surface of the robot where c is a positive scalar and the

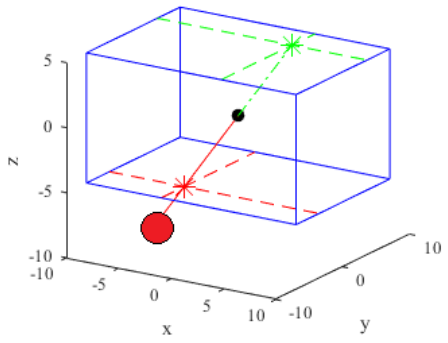

Figure 5.1: Diagram of maximum avoidance acceleration, $U_{k_{max}}$.

control surface is defined by $u_{i_{max}}$ in each direction. A visualization of $U_{k_{max}}$ is shown in Fig. 5.1 where it is the point on the control surface opposite the vector $\nabla\phi_k$. This figure shows an example of finding $U_{k_{max}}$ from the vector $-c\nabla\phi_k$ shown as the solid dark line. The box represents the control space created from $u_{i_{max}}$ in each direction around the robot represented by the dot at the origin. The green dash-dot line represents $U_{k_{max}}$, the maximum vector of acceleration opposite to the red line representing the vector $\nabla\Phi_k$ to the avoidance region represented by the red circle.

We can now find the *closest* obstacle point considering velocity and acceleration as $p_n = \min_{p_k \in \mathbb{P}}(d_k - \frac{\|\nu\|^2 \cos(\theta_k)}{2(\|U_{k_{max}}\|)})$, where $d_k = \|\nabla\phi_k\|$ is the distance to the obstacle point p_k and $\theta_k = \cos^{-1}(\frac{\nu \cdot \nabla\phi_k}{\|\nu\|\|\nabla\phi_k\|})$ is the angle between the direction of travel and the angle to face p_k. We define the obstacle vector as,

$$\nabla\phi_n = p - p_n = \begin{bmatrix} x \\ y \\ z \end{bmatrix} - \begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix}, \tag{5.1}$$

which will be used for determining safety of the robot. We also let $d_n = d_k$, $\theta_n = \theta_k$, and $U_{n_{max}} = U_{k_{max}}$ for the respective $p_k = p_n$ which is the *closest* obstacle point.

We introduce a switching signal $\zeta \in \{0, 1\}$ as,

$$\zeta = \begin{cases} 0 & \text{if } |\theta_n| \geq \frac{\pi}{2} \\ & \text{or } d_n > d_m \\ 1 & \text{else} \end{cases}, \tag{5.2}$$

where $d_m = \frac{\|\nu\|^2 \cos(\theta_n)}{2(\|U_{n_{max}}\| - \alpha_{max})}$ and $\alpha_{max} = \sqrt{\alpha_x^2 + \alpha_y^2 + \alpha_z^2}$ is the maximum magnitude of the input disturbance vector. The distance d_m represents the worst case distance required to accelerate the robot away from an obstacle, given the component of velocity ν towards the obstacle and the robots maximum acceleration in the opposite direction accounting for disturbance. Note that (5.2) states the robot is safe if the obstacle is far away or the obstacle is safely outside the robot's current direction of motion.

5.2 Tracking Vectors

The problem is formulated as tracking for each case of the switching signal as for the wheeled robot. In the case of safe operation, $\zeta = 0$, the robot tracks the safe vector,

$$\nabla\phi_s = (p_g|\dot{p}_g) - (p|\dot{p}) = \begin{bmatrix} x_g \\ y_g \\ z_g \\ \dot{x}_g \\ \dot{y}_g \\ \dot{z}_g \end{bmatrix} - \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}, \tag{5.3}$$

where p represents the robot and p_g is the goal with respective velocities.

For unsafe operation, $\zeta = 1$, the robot uses $(\nabla\phi_n|\nabla\dot{\phi}_n)$, the avoidance vector $\nabla\phi_n$ from (5.1) and its derivative.

Similar to the wheeled mobile robot, the unified tracking error which encompasses both the safe and unsafe operation modes is defined as,

$$\epsilon = \zeta(\nabla\phi_n|\nabla\dot{\phi}_n) + (1-\zeta)\nabla\phi_s = \begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \epsilon_z \\ \dot{\epsilon}_x \\ \dot{\epsilon}_y \\ \dot{\epsilon}_z \end{bmatrix} = \begin{bmatrix} x_r - x \\ y_r - y \\ z_r - z \\ \dot{x}_r - \dot{x} \\ \dot{y}_r - \dot{y} \\ \dot{z}_r - \dot{z} \end{bmatrix}, \tag{5.4}$$

where the subscript r denotes the reference in either case of safe or unsafe operation. This unified tracking error is used for the control law development and proof of stability. We note that in the safe operation case, the vector is defined as the error between the robot and the goal. In the unsafe operation case the vector is defined to drive the robot towards an equilibrium on the avoidance

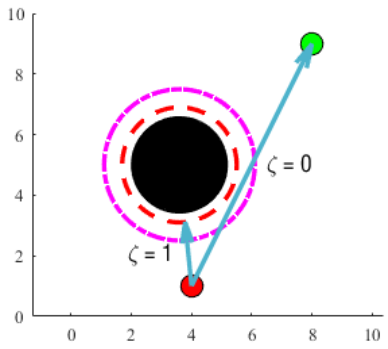

Figure 5.2: Diagram of error, ϵ, for safe and unsafe case.

region boundary and avoid collision. In the case of safe operation goal velocities of the quad-rotor are directly provided, either from a moving target which the robot is following, or 0 for a static goal. In the unsafe case the velocities are calculated by the rate of change of the closest point on the avoidance region, p_n, which will be the component of the quad-rotor velocity tangential to the obstacle. A diagram of ϵ in each case is shown in Fig. 5.2. The small dark circle is the robot and the small light circle is the goal. The large black circle represents the obstacle and the red dashed line is the avoidance region. The magenta dash-dot line is the varying unsafe region depending on the robots position and velocity. The vectors show the gradient to the reference point in each case. The diagram shows how the robot chooses a reference at the goal or the avoidance region of the obstacle depending on safety conditions. The controller ensures the robot never enters the region within constant radius r of the obstacle.

5.3 Error Dynamics

The control law is formulated with the objective of reaching a virtual target similar to the wheeled robot but with the addition of virtual velocities. The target is determined by adding the unified tracking error vector, ϵ (5.4), to the robot state $(p|\dot{p})$ resulting in the reference $(p_r|\dot{p}_r)$. Since this model does not consider rotation, the unified vector is the error signal and does not need to be translated.

Taking the derivative of the error signal we get,

$$\dot{\epsilon} = \begin{bmatrix} \dot{\epsilon}_x \\ \dot{\epsilon}_y \\ \dot{\epsilon}_z \\ \ddot{\epsilon}_x \\ \ddot{\epsilon}_y \\ \ddot{\epsilon}_z \end{bmatrix} = \begin{bmatrix} \dot{x}_r - \dot{x} \\ \dot{y}_r - \dot{y} \\ \dot{z}_r - \dot{z} \\ \ddot{x}_r - \ddot{x} \\ \ddot{y}_r - \ddot{y} \\ \ddot{z}_r - \ddot{z} \end{bmatrix} = \begin{bmatrix} \dot{x}_r - \dot{x} \\ \dot{y}_r - \dot{y} \\ \dot{z}_r - \dot{z} \\ \ddot{x}_r - u_x + \delta_x \\ \ddot{y}_r - u_y + \delta_y \\ \ddot{z}_r - u_z + \delta_z \end{bmatrix}. \tag{5.5}$$

5.4 Control Laws And Stability Analysis

For stability analysis of the control system we follow a similar approach as the wheeled robot and formulate the error as a hybrid system,

$$\dot{\epsilon} = \begin{cases} f_1 & \text{if } \zeta = 0 \\ f_2 & \text{else} \end{cases}, \tag{5.6}$$

which makes subsystems ,f_1 and f_2, of the error derivative (5.5) for each case of the switching signal (5.2), either safe ($\zeta = 0$) or unsafe ($\zeta = 1$).

We propose the Lyapunov candidate function,

$$V(\epsilon) = \frac{1}{2}(\epsilon_x^2 + \epsilon_y^2 + \epsilon_z^2 + \dot{\epsilon}_x^2 + \dot{\epsilon}_y^2 + \dot{\epsilon}_z^2). \tag{5.7}$$

This candidate function is of differentiability class C^1 and is positive everywhere but the origin, $\epsilon = 0$ where $V(0) = 0$. Taking the derivative of the Lyapunov candidate function we get,

$$\begin{aligned} \dot{V}(\epsilon) =& \epsilon_x\dot{\epsilon}_x + \epsilon_y\dot{\epsilon}_y + \epsilon_z\dot{\epsilon}_z + \dot{\epsilon}_x\ddot{\epsilon}_x + \dot{\epsilon}_y\ddot{\epsilon}_y + \dot{\epsilon}_z\ddot{\epsilon}_z \\ =& \epsilon_x\dot{\epsilon}_x + \epsilon_y\dot{\epsilon}_y + \epsilon_z\dot{\epsilon}_z + \dot{\epsilon}_x(u_{r_x} - u_x + \delta_x) + \dot{\epsilon}_y(u_{r_y} - u_y + \delta_y) + \dot{\epsilon}_z(u_{r_z} - u_z + \delta_z). \end{aligned} \tag{5.8}$$

Like the position and velocity, the reference acclerations u_{r_i} are either known from the target being tracked or measured as the second time derivative of the avoidance vector.

A controller will be developed for each case of the hybrid system, either f_1 or f_2, so that the final controller will be of the form,

$$U = \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} = \zeta U_n + (1-\zeta)U_s = \zeta \begin{bmatrix} u_{n_x} \\ u_{n_y} \\ u_{n_z} \end{bmatrix} + (1-\zeta) \begin{bmatrix} u_{s_x} \\ u_{s_y} \\ u_{s_z} \end{bmatrix}, \tag{5.9}$$

where ζ is the safety switching signal (5.2).

5.4.1 Control Law: Unsafe

We will first examine the unsafe case ($\zeta = 1$). The control in this case is $U = U_n = U_{n_{max}}$ where $U_{n_{max}}$ is the maximum acceleration of the robot away from the obstacle developed in section 5.1. As the safety condition was considering the worst case scenario with this control, it will guarantee that collision with the obstacle is avoided. We must then prove stability. Inserting the control inputs into the Lyapunov candidate derivative (5.8), the Lyapunov candidate derivative becomes,

$$\begin{aligned} \dot{V}(\epsilon) =& \dot{\epsilon}_x(\epsilon_x + \ddot{x}_r - u_{n_x} + \delta_x) + ... \\ & \dot{\epsilon}_y(\epsilon_y + \ddot{y}_r - u_{n_y} + \delta_y) + ... \\ & \dot{\epsilon}_z(\epsilon_z + \ddot{z}_r - u_{n_z} + \delta_z). \end{aligned} \tag{5.10}$$

Letting $\epsilon_\nu := [\dot{\epsilon}_x, \dot{\epsilon}_y, \dot{\epsilon}_z]$ and $\delta := [\delta_x, \delta_y, \delta_z]$, the Lyapunov derivative (5.10) can be re-written,

$$\begin{aligned} \dot{V}(\epsilon) =& \dot{\epsilon}_x(\epsilon_x + \ddot{x}_r) + \dot{\epsilon}_y(\epsilon_y + \ddot{y}_r) + \dot{\epsilon}_z(\epsilon_z + \ddot{z}_r) + ... \\ & - \epsilon_\nu \cdot U_{n_{max}} + \epsilon_\nu \cdot \delta. \end{aligned} \tag{5.11}$$

Since the reference in the unsafe case is defined as the closest obstacle point on the perimeter of the avoidance region from the robot, the velocity of the reference is the component of the robot velocity tangential to $\nabla\phi_o$. The velocity error ϵ_ν is therefore the robot's velocity component away from the obstacle and in the direction of $-\nabla\phi_o$. The acceleration of the robot $U_{n_{max}}$ is in the the direction of $-\nabla\phi_o$, leaving no tangential component and therefore $\ddot{i}_r = 0$. This makes $\dot{\epsilon}_i(\epsilon_i + \ddot{i}_r)$ negative. As

$U_{n_{max}}$ and ϵ_ν are in the same direction, $-\epsilon_\nu U_{n_{max}} = \|\epsilon_\nu\|\|U_{n_{max}}\|\cos(\pi) = -\|\epsilon_\nu\|\|U_{n_{max}}\|$. Small perturbations from δ result in $\psi = \pi + \angle(U_{n_{max}} - \delta)$ such that $sgn(\|\epsilon_\nu\|\|U_{n_{max}}\|\cos(\psi))$ is negative. The Lyapunov derivative is negative definite as long as $\|U_{n_{max}}\| > \alpha_{max}$, the maximum unsafe control magnitude is always greater than the maximum disturbance vector. This is a reasonable assumption because it would otherwise imply disturbance greater than the maximum control effort, which would render most systems useless. This proves the system is asymptotically stable in the unsafe case.

5.4.2 Control Law: Safe

This section will develop control laws for the safe case ($\zeta = 0$). The control in this case is $U = U_s$. The controller will be analyzed for simplified versions of the problem without disturbance or saturation first and then expanded upon for the model including input disturbance and input saturation used in experiments.

5.4.2.1 No Disturbance or Saturation

This section develops a control law considering error dynamics from (5.5) and the Lyapunov candidate function derivative (5.8) with $\delta_i = 0$ and no input saturation, $u_{i_{max}} = \infty$. The control inputs are chosen as,

$$u_i = \epsilon_i + \ddot{i}_r + \dot{\epsilon}_i. \tag{5.12}$$

Putting these control inputs into the Lyapunov candidate derivative (5.8) results in,

$$\dot{V}(\epsilon) = -\dot{\epsilon}_x^2 - \dot{\epsilon}_y^2 - \dot{\epsilon}_z^2. \tag{5.13}$$

This Lyapunov derivative is monotonically decreasing and therefore shows that the system is stable. This is not yet sufficient to determine asymptotic convergence as the derivative, $\dot{V}(\epsilon)$, is only negative semi-definite and is zero when $\dot{\epsilon}_x = \dot{\epsilon}_y = \dot{\epsilon}_z = 0$, without considering position error. Asymptotic stability is shown through the invariant set theorem [41], showing the system will converge to the largest invariant set M within the set R of all states where $\dot{V}(\epsilon) = 0$. The set R

contains the origin, which is also in the set M as it is an equilibrium and thus an invariant set. Consider state $s \in R$ outside of the origin. To remain at this state, the velocity of the robot must remain equal to the reference. With $\dot{\epsilon}_i = 0$ and $|\epsilon_i| > 0$, $u_i = \ddot{i}_r + \epsilon_i \neq \ddot{i}_r$. This shows that the robot will immediately leave these states as the reference accelerates at a different rate than the robot, a contradiction for an invariant set. This shows that the invariant set M only contains the origin and the system is asymptotically stable. This is true for the entire state space and $V(\epsilon) \to \infty$ as $\|\epsilon\| \to \infty$, proving the system is globally asymptotically stable.

5.4.2.2 Bounded Disturbance

Consider error dynamics given in (5.5) including disturbances $\delta_i \leq \alpha_i$ and no input saturation, $u_{i_{max}} = \infty$. The control inputs are chosen as,

$$u_i = \epsilon_i + \ddot{i}_r + \dot{\epsilon}_i + \dot{\epsilon}_i \alpha_i, \tag{5.14}$$

with the disturbance rejection term, $\dot{\epsilon}_i \alpha_i$, added to the simplified case (5.12). Inserting the new controls into the Lyapunov candidate derivative (5.8), the Lyapunov candidate derivative becomes,

$$\begin{aligned} \dot{V}(\epsilon) =& \dot{\epsilon}_x(\delta_x - \dot{\epsilon}_x - \dot{\epsilon}_x\alpha_x) + \dot{\epsilon}_y(\delta_y - \dot{\epsilon}_y - \dot{\epsilon}_y\alpha_y) + ... \\ & \dot{\epsilon}_z(\delta_z - \dot{\epsilon}_z - \dot{\epsilon}_z\alpha_z) \\ =& -\dot{\epsilon}_x^2 + \dot{\epsilon}_x\delta_x - \dot{\epsilon}_x^2\alpha_x - \dot{\epsilon}_y^2 + \dot{\epsilon}_y\delta_y - \dot{\epsilon}_y^2\alpha_y - ... \\ & \dot{\epsilon}_z^2 + \dot{\epsilon}_z\delta_z - \dot{\epsilon}_z^2\alpha_z. \end{aligned} \tag{5.15}$$

Since each disturbance δ_i is bounded by α_i, $\dot{\epsilon}_z\delta_z \leq |\dot{\epsilon}_i|\alpha_i$. The disturbance and disturbance rejection is now upper bounded as $\dot{\epsilon}_i\delta_i - \dot{\epsilon}_i^2\alpha_i \leq |\dot{\epsilon}_i|\alpha_i(1 - |\dot{\epsilon}_i|)$. The maximum value of this function is at $|\dot{\epsilon}_i| = \frac{1}{2}$ resulting in a maximum value of $\frac{1}{4}\alpha_i$. The Lyapunov derivative (5.15) can be written,

$$\dot{V}(\epsilon) \leq -\dot{\epsilon}_x^2 - \dot{\epsilon}_y^2 - \dot{\epsilon}_z^2 + \frac{1}{4}(\alpha_x + \alpha_y + \alpha_z). \tag{5.16}$$

This has provided a bound for the negative semi-definiteness of the Lyapunov derivative proving the system is GUUB. We can now find what the bound is. The magnitude of the velocity error is $\sqrt{\dot{\epsilon}_x^2 + \dot{\epsilon}_y^2 + \dot{\epsilon}_z^2}$ and the square of this must be greater than $\frac{1}{4}(\alpha_x + \alpha_y + \alpha_z)$ to ensure the function is negative definite. Therefore a magnitude of velocity error greater than $\frac{1}{2}(\sqrt{\alpha_x + \alpha_y + \alpha_z})$ ensures the function is negative definite. This proves that all states will asymptotically converge to within this velocity error region.

A bound on the position error can be found through the velocity bound. For $\ddot{\epsilon}_i = \dddot{i}_r - u_i$, substitute the control input $u_i = \dot{\epsilon}_i + \epsilon_i + \ddot{i}_r + \dot{\epsilon}_i\alpha_i$ to obtain $\ddot{\epsilon}_i = -\dot{\epsilon}_i(1 + \alpha_i) - \epsilon_i$. It is clear that $\text{sgn}(\epsilon_i) = -\text{sgn}(\ddot{\epsilon}_i)\ \forall\ |\epsilon_i| > |\dot{\epsilon}_i(1 + \alpha_i)|$. In other words, for all states which satisfy $|\epsilon_i| > |\dot{\epsilon}_i(1 + \alpha_i)|$, the acceleration will be in the direction to reduce position error. As the velocity error has already been bounded, the position bound can be updated as $\frac{1}{2}(\sqrt{\alpha_x + \alpha_y + \alpha_z})(1 + \alpha_i)$.

5.4.2.3 Input Saturation

Consider error dynamics given in (5.5) without disturbances $\delta_i = 0$ and input saturation with a maximum acceleration of $u_{i_{max}}$ for the robot and $\dddot{i}_{r_{max}}$ for the target. The control inputs are chosen as,

$$u_i = \sigma(\epsilon_i + \ddot{i}_r + \dot{\epsilon}_i), \tag{5.17}$$

where $\sigma(x)$ is the saturation function,

$$\sigma(x) = \begin{cases} x_{max} & \text{if } (x > x_{max}) \\ -x_{max} & \text{if } (x < -x_{max}) \\ x & \text{else} \end{cases}. \tag{5.18}$$

Inserting these controls, the Lyapunov candidate derivative (5.8) becomes,

$$\begin{aligned} \dot{V}(\epsilon) =& \dot{\epsilon}_x(\epsilon_x + \ddot{x}_r - \sigma(\dot{\epsilon}_x + \epsilon_x + \ddot{x}_r)) + ... \\ & \dot{\epsilon}_y(\epsilon_y + \ddot{y}_r - \sigma(\dot{\epsilon}_y + \epsilon_y + \ddot{y}_r)) + ... \\ & \dot{\epsilon}_z(\epsilon_z + \ddot{z}_r - \sigma(\dot{\epsilon}_z + \epsilon_z + \ddot{z}_r)). \end{aligned} \tag{5.19}$$

For the case when the saturation value is not reached, (5.13) applies and the system is asymptotically stable. The interesting case is when the acceleration is saturated, $|\dot{\epsilon}_i+\epsilon_i+\dddot{i}_r| > u_{i_{max}}$. Within this case there are sub-cases for $sgn(\dot{\epsilon}_i) = -sgn(\epsilon_i + \dddot{i}_r)$ referred to as case A, and $sgn(\dot{\epsilon}_i) = sgn(\epsilon_i+\dddot{i}_r)$ referred to as case B. For stability in both cases $sgn(\dot{\epsilon}_i) = -sgn(\epsilon_i+\dddot{i}_r-\sigma(\dot{\epsilon}_i+\epsilon_i+\dddot{i}_r))$ must be true, referred to as the stability condition. This will ensure that the Lyapunov derivative is negative definite.

For case A $sgn(\epsilon_i+\dddot{i}_r) = sgn(\epsilon_i+\dddot{i}_r-\sigma(\dot{\epsilon}_i+\epsilon_i+\dddot{i}_r))$ satisfies the stability condition because the acceleration does not change the sign of $\epsilon_i + \dddot{i}_r$. Since $sgn(\dot{\epsilon}_i) = -sgn(\epsilon_i + \dddot{i}_r)$ in case A, $\sigma(\dot{\epsilon}_i + \epsilon_i + \dddot{i}_r)$ is always a smaller magnitude or opposite sign of $\epsilon_i + \dddot{i}_r$. Therefore subtracting $\sigma(\dot{\epsilon}_i + \epsilon_i + \dddot{i}_r)$ from $\epsilon_i + \dddot{i}_r$ will not change the sign and the stability condition is always met in case A.

In case B we know that $sgn(\dot{\epsilon}_i) = sgn(\epsilon_i + \dddot{i}_r)$ and therefore $sgn(\dot{\epsilon}_i) = sgn(\dot{\epsilon}_i + \epsilon_i + \dddot{i}_r) = sgn(u_i)$. Considering the saturated case $u_i = u_{i_{max}} sgn(\dot{\epsilon}_i)$. We can now write $\dot{\epsilon}_i$ as,

$$\dot{\epsilon}_i = \dot{\epsilon}_{i_o} + \int (\dddot{i}_r - sgn(\dot{\epsilon}_i) u_{i_{max}}) dt. \tag{5.20}$$

From this we can conclude that from a bounded initial velocity error, $\dot{\epsilon}_{i_o}$ with the condition $|\dddot{i}_r| < u_{i_{max}}$, case B will always return to either case A or the unsaturated case in finite time. We now must examine returning to case B. The acceleration $u_i = \sigma(\epsilon_i+\dddot{i}_r+\dot{\epsilon}_i)$ must either change $sgn(\dot{\epsilon}_i)$ or $sgn(\epsilon_i + \dddot{i}_r)$ to return to case B. We look at the case where $sgn(\dot{\epsilon}_i)$ changes. This requires that $sgn(\dddot{i}_r - u_i) = -sgn(\dot{\epsilon}_i)$. As the only acceleration term which may change $sgn(\dot{\epsilon}_i)$ comes from $\dot{\epsilon}_i$, as $|\dot{\epsilon}_i| \to 0$, $sgn(\dddot{i}_r - u_i) = sgn(\dot{\epsilon}_i)$ and $sgn(\dot{\epsilon}_i)$ will not change. We then look at the case where $sgn(\epsilon_i + \dddot{i}_r)$ changes. Since $\dddot{i}_r$ is upper bounded by $\dddot{i}_r \leq \dddot{i}_{r_{max}}$, we know that at the time of switching signs $\epsilon_i \leq \dddot{i}_{r_{max}}$. This is therefore the only way to re-enter case B, referred to as the subset of case B denoted B', which occurs when the target is overshot. The stability condition in case B is satisfied by $sgn(\epsilon_i + \dddot{i}_r) = -sgn(\epsilon_i + \dddot{i}_r - \sigma(\dot{\epsilon}_i + \epsilon_i + \dddot{i}_r))$. This requires $|\sigma(\dot{\epsilon}_i+\epsilon_i+\dddot{i}_r)| > |\epsilon_i+\dddot{i}_r|$ which becomes $u_{i_{max}} > |\epsilon_i+\dddot{i}_r|$ because we are considering the saturated

acceleration. To prove asymptotic stability in case B' we substitute the upper bound on ϵ_i to get $u_{i_{max}} > 2\ddot{i}_{r_{max}}$.

With sufficient maximum acceleration $u_{i_{max}} > 2\ddot{i}_{r_{max}}$ and bounded initial velocity error we guarantee,

1. The unsaturated case is asymptotically stable
2. Case A is asymptotically stable
3. Case B' is asymptotically stable
4. The system will only enter case B\B' from initial conditions
5. The system will leave case B in finite time ,

which together prove the system is asymptotically stable.

5.4.2.4 Bounded Disturbance and Input Saturation

Consider error dynamics given in (5.5) with disturbances $\delta_i \leq \alpha_i$ and input saturation with a maximum acceleration of $u_{i_{max}}$ for the robot and $\ddot{i}_{r_{max}}$ for the target. The control inputs are chosen as,

$$u_i = \sigma(\epsilon_i + \ddot{i}_r + \dot{\epsilon}_i + \dot{\epsilon}_i\alpha_i). \tag{5.21}$$

Inserting these controls, the Lyapunov candidate derivative (5.8) becomes,

$$\begin{aligned}\dot{V}(\epsilon) =& \dot{\epsilon}_x(\epsilon_x + \ddot{x}_r + \delta_x - \sigma(\dot{\epsilon}_x(1-\alpha_x) + \epsilon_x + \ddot{x}_r)) + ... \\ & \dot{\epsilon}_y(\epsilon_y + \ddot{y}_r + \delta_y - \sigma(\dot{\epsilon}_y(1-\alpha_y) + \epsilon_y + \ddot{y}_r)) + ... \\ & \dot{\epsilon}_z(\epsilon_z + \ddot{z}_r + \delta_z - \sigma(\dot{\epsilon}_z(1-\alpha_z) + \epsilon_z + \ddot{z}_r)).\end{aligned} \tag{5.22}$$

For the case when the saturation value is not reached, (5.16) applies and the system globally converges to a bounded region. When the acceleration is saturated there is again two cases. Case A is when $sgn(\dot{\epsilon}_i) = -sgn(\epsilon_i + \ddot{i}_r + \delta_i)$ and case B is $sgn(\dot{\epsilon}_i) = sgn(\epsilon_i + \ddot{i}_r + \delta_i)$. In case

A the Lyapunov derivative is negative definite when $sgn(\epsilon_i + \ddot{i}_r + \delta_i) = sgn(\epsilon_i + \ddot{i}_r + \delta_i - \sigma(\dot{\epsilon}_i(1+\alpha_i) + \epsilon_i + \ddot{i}_r))$. This is always satisfied for $|\dot{\epsilon}_i| > \frac{\alpha_i}{1+\alpha_i}$. We follow the same approach as in the case without disturbance and denote B' as the only region of B which the system will re-enter with $\epsilon_i \leq \ddot{i}_{r_{max}} + \alpha_i$. In case B it is required that $sgn(\epsilon_i + \ddot{i}_r + \delta_i) = -sgn(\epsilon_i + \ddot{i}_r + \delta_i - \sigma(\dot{\epsilon}_i(1+\alpha_i) + \epsilon_i + \ddot{i}_r))$ for the Lyapunov derivative to be negative definite. To ensure this $|\sigma(\dot{\epsilon}_i(1+\alpha_i) + \epsilon_i + \ddot{i}_r)| = u_{i_{max}} > |\epsilon_i + \ddot{i}_r + \delta_i|$ must be true. This is always satisfied for the same condition as in case A, $|\dot{\epsilon}_i| > \frac{\alpha_i}{1+\alpha_i}$ which can be translated to a position bound of $|\epsilon_i| \leq \alpha_i$ using the same approach as (5.16), as well as an additional condition of the max acceleration, $u_{i_{max}} > 2(\ddot{i}_{rmax} + \alpha_i)$.

Following the no disturbance case with sufficient maximum acceleration $u_{i_{max}} > 2(\ddot{i}_{r_{max}} + \alpha_i)$ and bounded initial velocity error the system is UUB.

CHAPTER 6

QUAD-ROTOR EXPERIMENTS

This chapter looks at the simulation and implementation results of the quad-rotor shared controller from (5.21). We will first implement the controller in Matlab with simulated disturbance and obstacles. The controller will then be tested on an actual quad-rotor.

6.1 Simulation

Simulations were run on Matlab for a robot with a maximum acceleration of $[u_{x_{max}}, u_{y_{max}}, u_{z_{max}}] = [3, 3, 5](\frac{m}{s^2})$. The velocity is also cut off at a maximum of $[v_{x_{max}}, v_{y_{max}}, v_{z_{max}}] = [4, 4, 6](\frac{m}{s})$ for practical and safety reasons for later implementation. The constants bounding disturbance in each direction are $\alpha_x = \alpha_y = 0.1\frac{m}{s^2}$ and $\alpha_z = 0.05\frac{m}{s^2}$. The avoidance region radius is defined as $r = 0.2m$. The first simulation was done for a a stationary goal at $[x_g, y_g, z_g] = [17, 20, 10](m)$ with the robot starting at $[x_0, y_0, z_0] = [0, 0, 0](m)$ with no initial velocity. The obstacle is a sphere located at $[x, y, z] = [10, 10, 9](m)$ with a radius of $3m$.

Fig. 6.1 shows the simulation of the quad-rotor model navigating to the stationary goal location while safely avoiding the obstacle in its path. An isometric view of the robot path is shown on the left, the path components are in the middle, and the Lyapunov function value over time is shown on the right. The initial robot position is shown as the blue asterisk and the end position is shown as the magenta asterisk. Dashed lines in the Lyapunov plot indicate the switching signal. The safe controller provides a trajectory which would cause collision with the object. The unsafe control takes over when the safety conditions are not met to divert the robot away from the obstacle. Once the robot meets safety conditions, the safe control resumes and the robot reaches the goal location. The robot senses an unsafe region and accelerates around the obstacle until reaching a safe point where it continues to track towards the goal. In the component plane view in Fig. 6.1 the obstacle

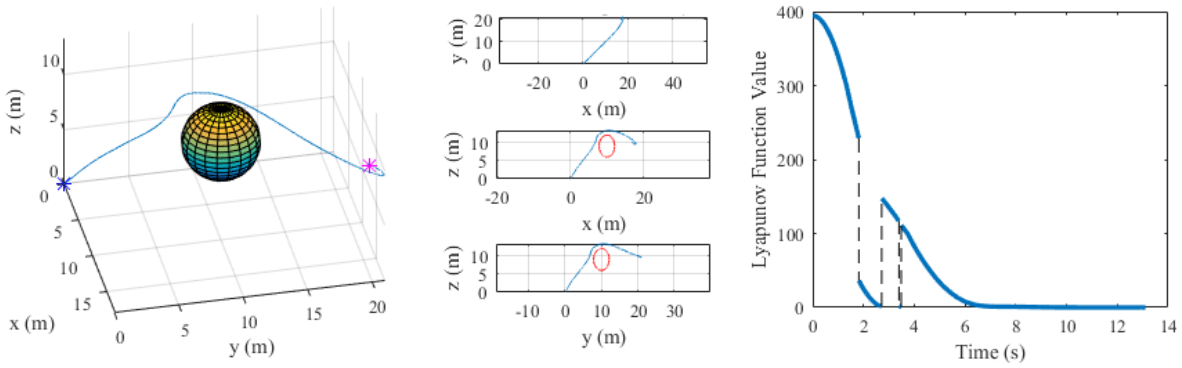

Figure 6.1: Quad-rotor simulation to track stationary goal with single sphere obstacle: path and Lyapunov value.

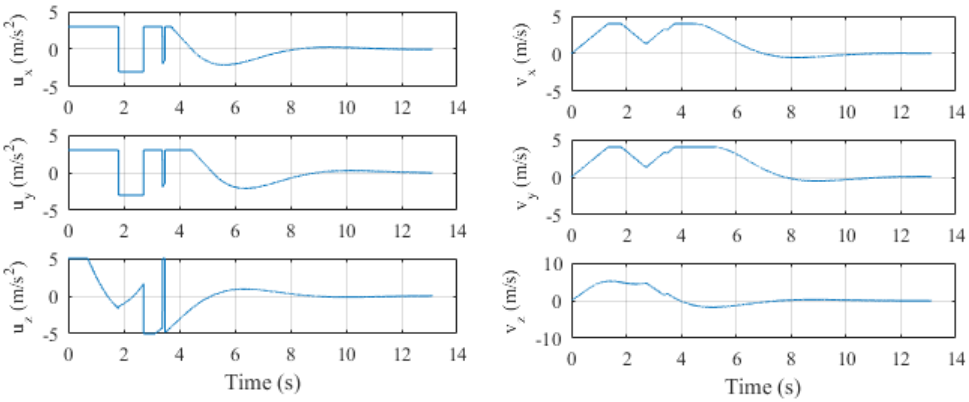

Figure 6.2: Quad-rotor simulation to track stationary goal with single sphere obstacle: accelerations and velocities.

is plotted as the points corresponding to a slice of the obstacle in the current plane at the robot's position in time. This component view shows that the avoidance region is maintained at all time. The Lyapunov function shows the error of the system is non-increasing at all times when the system is not switching and that the robot is able to reach the goal eventually.

Fig. 6.2 shows the accelerations on the left and velocities on the right of each component of the quad-rotor model during the simulation. These figures show the saturation of the acceleration and velocity of the robot. It is clear that although the robot is reaching maximum acceleration and velocity it always stays out of the avoidance region around the obstacle and reaches the goal.

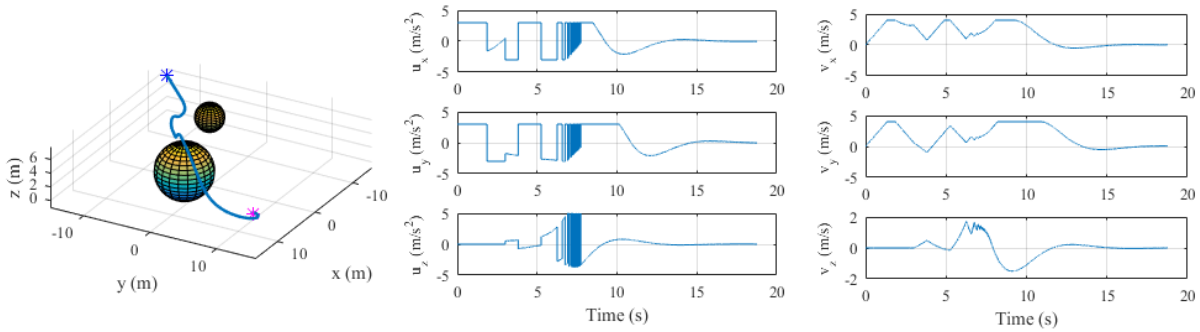

Figure 6.3: Quad-rotor simulation to track stationary goal through obstacle field: path, accelerations, and velocities.

The next simulation was done for a robot starting at $[x_0, y_0, z_0] = [-15, -15, 5](m)$ and traversing to a stationary goal location at $[x_g, y_g, z_g] = [15, 15, 5](m)$ through an obstacle field. There are two obstacles along the path, one at $[x, y, z] = [-7, -4, 5](m)$ with a radius of $2m$ and the other at $[x, y, z] = [5, -1, 3](m)$ with a radius of $4m$.

The simulation results are shown in Fig.6.3. An isometric view of the robot path is on the left, the robot accelerations are shown in the middle, and robot velocities are shown on the right. The initial robot position is shown as the blue asterisk and the end position is shown as the magenta asterisk. The unsafe regions cause the robot to accelerate tangentially from the obstacles in the direction that allows the robot to stay closest to its desired path. Throughout the trajectory the robot maintains a minimum distance from the obstacles and does not enter the avoidance region. The velocity and acceleration plots show the robot's response to obstacles detected throughout traversal.

6.2 Implementation

The shared control algorithm was implemented with Quanser Autonomous Vehicle Research Studio [38] for the Q-Drone shown in Fig. 6.4 which uses motion motion capture cameras for state estimation. The markers used for camera tracking are circled in red. The system used for motion tracking of the Q-Drone is Optitrack Motion Capture Systems [36] which uses 6 Flex 13 cameras in a netted $[L, W, H] = [4.5, 4.5, 2.5](m)$ enclosure which corresponds to a

Figure 6.4: Quanser Autonomous Vehicle Research Studio Q-Drone.

$[L, W, H] = [3.5, 3.5, 2.0](m)$ work space for the Q-Drone. The calibrated motion capture system is capable of sub-millimeter accuracy in position tracking of the markers on the Q-Drone.

Given a goal position, $p_g = [x_g, y_g, z_g]$, and an estimation of the robot position, $p = [x, y, z]$, the shared control sends a desired position, $p_d = [x_d, y_d, z_d]$, from a ground station to the Q-Drone. The Q-Drone then uses an on-board Proportional Integral Velocity (PIV) controller from Quanser to drive the Q-Drone to the desired state using the rotational model of the quad-rotor. As the desired state only includes position, the remaining states of the robot are used to control the Q-Drone to that position. The motion capture then sends position estimation back to the ground control station. The Q-Drone also sends inertial measurement unit (IMU) measurements back to the ground station. The are used in conjunction to estimate the full state of the Q-Drone. Rotor commands are sent to the Q-Drone via a ground station.

The control diagram is shown if Fig. 6.5. This diagram shows how the shared control is used with the Quanser controller to operate the Q-Drone. For these experiments the obstacles were defined in code and measurements of the obstacles from the quad-rotor where simulated. Parameters from Quanser for the Q-Drone are a total mass of $m = 1.121kg$ and moments of inertia of $[I_{xx}, I_{yy}, I_{zz}] = [0.01, 0.0082, 0.0148](kgm^2)$. The force and moment constants are $K_F = 5.11N$ and $K_M = 0.0487Nm$. The size of the Q-Drone is $[L, W, H] = [0.363, 0.403, 0.139](m)$.

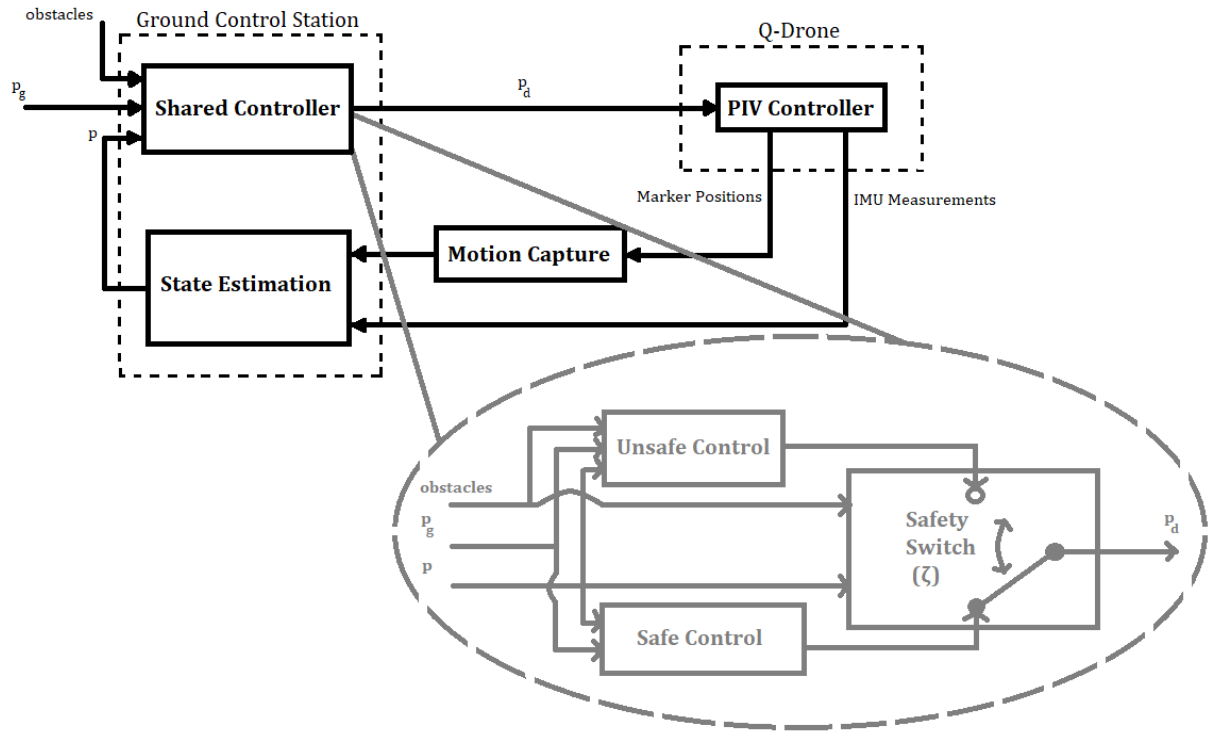

Figure 6.5: Control diagram for quad-rotor experiments.

The first experiment with the Q-Drone is tracking a target moving in a circular pattern. The goal trajectory used in the experiment is a $0.5m$ high circle with $1m$ radius around the origin which the reference travels around at $0.4\frac{rad}{s}$. A spherical object with $0.1m$ radius is simulated at $[0, 1, 0.5](m)$ for the robot to avoid during tracking. The maximum accelerations of the robot are $[u_{x_{max}}, u_{y_{max}}, u_{z_{max}}] = [1, 1, 1.5](\frac{m}{s^2})$ and the maximum velocities are $[v_{x_{max}}, v_{y_{max}}, v_{z_{max}}] = [0.4, 0.4, 0.5](\frac{m}{s})$ as the space is not as large as previous simulations. The acceleration error is bounded by $[\alpha_x, \alpha_y, \alpha_z] = [0.1, 0.1, 0.2](m)$ and the avoidance constant is $r = 0.25m$ to account for the largest dimension of the Q-Drone.

The experimental results of the quad-rotor tracking circular motion is shown in Fig. 6.6. An isometric view of the experimental path of the robot is shown on the left. The black dashed line shows the reference trajectory being tracked. The blue dash-dot line shows the output of the shared controller for the quad-rotor to track with a circle marker at the start and an asterisk at the end of the trajectory. The red lines show Optitrack data of the actual robot path. The path components are

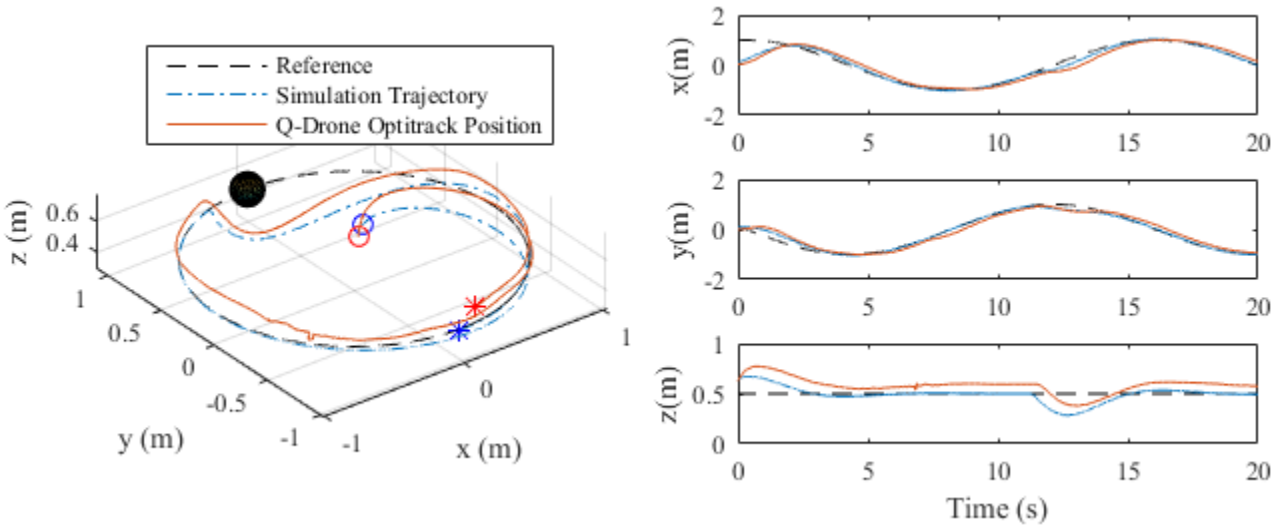

Figure 6.6: Quad-rotor experiment to track circular planar motion with single sphere obstacle: path and components.

shown on the right. The experimental data shows the robot is able to effectively track a moving trajectory while avoiding obstacles. There is a consistent difference in the z component between the shared control output and the Optitrack data after take-off, this is due to the physical height of the robot. Although the dynamics used in the inner loop are simplified from actual quad-rotor dynamics, the double integrator system is a close enough approximation for the quad-rotor to safely avoid collision with obstacles. The position components are also shown in Fig. 6.6. The components show that the robot is able to closely track the simulated trajectory and avoid obstacles.

The application of this control was also experimentally shown for a quad-rotor navigating through an rectangular obstacle field in Fig. 6.7. The parameters of the system are the same as in the previous experiment and the robot traverses from a start location of $(-1.5, -1.5, 0.5)$ to a goal location of $(1.5, 1.5, 0.5)$ while passing two rectangular structures.The experimental setup of the obstacle field for quad-rotor navigation is shown on the left with the quad-rotor shown for scale at position $[x, y, z] = [0, 0, 0](m)$. The experimental path of robot is shown on the right. The blue line shows the shared control inner loop output to the quad-rotor with the circle marker at the start and the asterisk at the end of the trajectory. The red lines show the Optitrack data of the actual

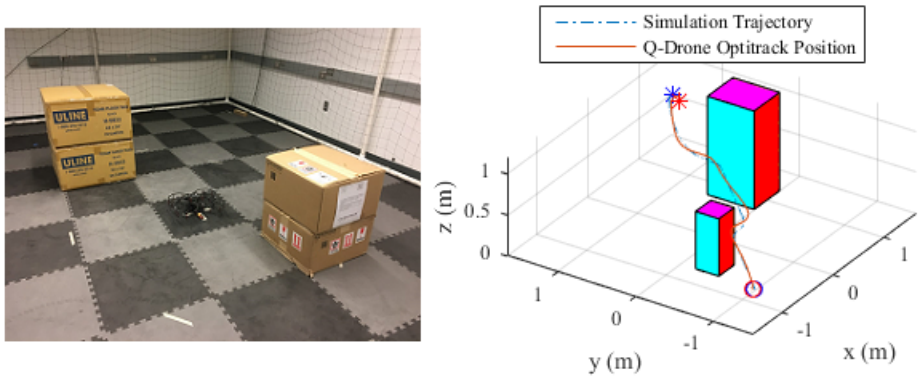

Figure 6.7: Quad-rotor experiment to track stationary goal with rectangular obstacle field: setup and path.

robot path. The robot was successful in avoiding obstacles and reaching the goal location. As the robot approached each obstacle the shared control took over to maintain the avoidance region around the obstacle. Inertia of the system returned the robot to safe operating conditions so it was able to divert around the obstacle and reach the goal.

CHAPTER 7

CONCLUSIONS

This chapter will discuss a summary and findings of this research. We will then propose how the work could be extended in the future.

7.1 Research Summary

In this thesis shared control is developed as a method of obstacle avoidance for a non-holonomic wheeled robot and a quad-rotor. The control methods are proven to avoid obstacles and proven stable through Lyapunov analysis. Simulations and experiments are done to validate the control method and demonstrate its effectiveness in conjunction with other control methods.

A model is created for the kinematic model of a non-holonomic wheeled robot. The model is constrained from lateral movement via a no-slip condition. With this model a shared control architecture is developed which switches between safe tracking of a goal and obstacle avoidance. Two controllers are developed in this architecture, one for a sensor noise model and one for an input disturbance and saturation model. Both of these are proven stable through Lyapunov analysis for an arbitrarily switching system. Simulations of the wheeled robot show the effectiveness of the controller for avoiding obstacles. It also shows how the shared control method can easily compliment different control methods such as tele-operation and path planning.

A simplified translational dynamic model of a quad-rotor is created for a shared controller. A more complete dynamic model of the quad-rotor is also discussed as it is used in implementation. The controller is designed incrementally which guarantees obstacle avoidance and accounts for bounded input disturbance and saturated input. Simulations are done with the simplified model to show the effectiveness. The controller is then implemented to provide desired positions to a PIV controller using the more complex model. The experiments show the shared controller is effective

when implemented. Experiments also show the controller able to track moving and accelerating targets.

This method of switched-system shared control is very effective for obstacle avoidance. The shared control ensures all obstacles, whether planned or unplanned, will be avoided while accounting for dynamics, disturbance, and input saturation. Through Lyapunov analysis it has been shown that both the wheeled mobile robot control and quad-rotor control result in bounded error, subject to reasonable acceleration constraints for the quad-rotor.

Simulation shows the effectiveness and adaptability of shared control as a method of obstacle avoidance. Experiments with a quad-rotor validate that the simplified dynamic model is sufficient in implementation for providing a desired state to a pre-built controller.

7.2 Future Extensions

This work establishes a foundation for extensions of obstacle avoidance methods. Such work could include expanding this to other vehicle and including more complex dynamics of the current vehicles in the shared control architecture. This would create less error between simulation and implementation which allows the user to be less conservative in creation of avoidance regions. This may be necessary for problems where there is only very narrow solutions. The obstacle avoidance method could also be adapted to more directly include the profile of the robot. The current method considers a constant radius robot in all directions but this is rarely true for actual robot. Allowing considerations of the robot profile may enable finding narrow solutions or finding more optimal paths. This may also enable this architecture to be used for non-mobile robots with linkages where assuming a constant radius of the entire robot for obstacle avoidance renders the system useless. With work in these areas a shared control architecture could be useful and easily implemented for many robot applications.

BIBLIOGRAPHY

[1] Amazon robots. https://www.amazonrobotics.com/#/, 2020. Online; accessed June 2020.

[2] Catalog of earth satellites. https://earthobservatory.nasa.gov/features/OrbitsCatalog, 2020. Online; accessed June 2020.

[3] Custom quad-rotor. https://www.pinterest.com/pin/474144667003250337/, 2020. Online; accessed June 2020.

[4] Dji enterprises camera drone. https://www.dji.com/, 2020. Online; accessed June 2020.

[5] Emergency delivery drone. https://www.jems.com/2020/01/29/new-partnership-in-drone-delivery-system-for-human-organs/, 2020. Online; accessed June 2020.

[6] irobot roomba. https://www.irobot.com/roomba, 2020. Online; accessed June 2020.

[7] Nasa mars exploration rovers. https://mars.nasa.gov/mer/, 2020. Online; accessed June 2020.

[8] Turtlebot 2. https://www.turtlebot.com/turtlebot2/, 2020. Online; accessed June 2020.

[9] A. Bacciotti and L. Mazzi. An invariance principle for nonlinear switched systems. In *Systems Control Letters*, volume 54(11), page 1109–1119, 2005.

[10] A. Benloucif, A. Nguyen, C. Sentouh, and J. Popieul. Cooperative trajectory planning for haptic shared control between driver and automation in highway driving. *IEEE Transactions on Industrial Electronics*, 66(12):9486–9857, 2019.

[11] M. Branicky. Multiple lyapunov functions and other analysis tools for switched and hybrid systems. In *IEEE Transactions on Automation and Control*, volume 43, pages 475–482, 1998.

[12] R. DeCarlo, M. Branicky, S. Pettersson, and B. Lennartson. Perspectives and results on the stability and stabilizability of hybrid systems. In *Proceedings of IEEE*, volume 88, pages 1069–1082, 2000.

[13] L. Doitsidis, K. P. Valavanis, N. C. Tsourveloudis, and M. Kontitsis. A framework for fuzzy logic based uav navigation and control. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 4, pages 4041–4046, 2004.

[14] R. E.Korf. Real-time heuristic search. *Artificial Intelligence*, 42:189–211, 1990.

[15] F. Flemish, D. Abbink, M. Itoh, M-P.Pacaux-Lemoine, and G. Webel. Shared control is the sharp end of cooperation: Towards a common framework of joint action, shared control and human machine cooperation. In *International Federation of Automatic Control*, 2016.

[16] J. Gammell, S. Srinivasa, and T. Barfoot. Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 3067–3074, May 2015.

[17] H. Goldstein, C. P. Poole, and J. L. Safko. *Classical Mechanics*. Pearson, 2011.

[18] G. Graetz and G. Michaels. Robots at work. In *The Review of Economics and Statistics*, volume C, 2018.

[19] J. Hespanh. Uniform stability properties of switched linear systems: Extensions of lasalle's invariance principle. In *IEEE Transactions on Automation and Control*, volume 49, pages 470–482, 2004.

[20] R. D. Howe and Y. Matsuoka. Robotics for surgery. In *Annual Review of Biomedical Engineering*, volume 1, pages 211–240, 1999.

[21] J. Jiang and A. Astolfi. Shared-control for the kinematic model of a mobile robot. In *IEEE Conference on Decision and Control*, volume 53, 2014.

[22] J. Jiang and A. Astolfi. Shared-control for a uav operating in the 3d space. In *European Control Conference (ECC)*, 2015.

[23] J. Jiang, P. D. Franco, and A. Astolfi. Shared-control for the kinematic and dynamic models of a mobile robot. In *IEEE Transactions on Control Systems Technology*, volume 24, 2016.

[24] J. Jin, N. Gans, Y.-G. Kim, and S.-G. Wee. A switched-system approach to shared robust control and obstacle avoidance for mobile robots. In *ASME Dynamic System and Control Conference*, 2014.

[25] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *IJRR*, 30(7):846–894, 2011.

[26] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research (IJRR)*, 5:90–98, 1986.

[27] S. LaValle and J. Kuffner. Randomized kinodynamic planning. *IJRR*, 20(5):378–400, May 2001.

[28] Y. Li, Z. Littlefield, and K. E. Bekris. Asymptotically optimal sampling-based kinodynamic planning. *IJRR*, 35(5):528–564, 2016.

[29] D. Liberzon. *Switching in Systems and Control*. Birkhauser, 2003.

[30] D. Liberzon and A. Morse. Basic problems in stability and design of switched systems. In *IEEE Control Systems Magazine*, volume 19, pages 59–70, 1999.

[31] M. Likhachev, G. Gordon, and S. Thrun. Anytime A* with provable bounds on sub-optimality. In *Advances in Neural Information Processing System*, 2004.

[32] C. Masone, M. Mohammadi, P. Giordano, and A. Franchi. Shared planning and control for mobile robots with integral haptic feedback. *International Journal of Robotics Research (IJRR)*, 37:1395–1420, 2018.

[33] R. M. Murray, Z. Li, and S. S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[34] K. Naderi, J. Rajamäki, and P. Hämäläinen. Rt-rrt*: a real-time path planning algorithm based on rrt*. In *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games*, pages 113–118, 2015.

[35] N. Nilsson. Problem-solving methods in artificial intelligence. *McGraw-Hill*, 1971.

[36] OptiTrack. Motion capture systems. `https://www.https://optitrack.com//`, 2020. Online; accessed June 2020.

[37] M. R. Petry, A. P. Moreira, R. A. M. Braga, and L. P. Reis. Shared control for obstacle avoidance in intelligent wheelchairs. In *IEEE Conference on Robotics, Automation and Mechatronics*, pages 182–187, 2010.

[38] Quanser. Autonomous vehicles research studio. `https://www.quanser.com/`, 2020. Online; accessed June 2020.

[39] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In *2009 IEEE International Conference on Robotics and Automation*, pages 489–494, 2009.

[40] T. Sangyam, P. Laohapiengsak, W. Chongcharoen, and I. Nilkhamhang. Path tracking of uav using self-tuning pid controller based on fuzzy logic. In *Proceedings of SICE Annual Conference 2010*, pages 1265–1269, 2010.

[41] J.-J. E. Slotine and W. Li. *Applied Nonlinear Control*. Prentice Hall, 1991.

[42] A. Stentz. The focussed D* algorithm for real-time replanning. In *In Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1652—-1659, 1995.

[43] C. Tingbin and Z. Qisong. Robot motion planning based on improved artificial potential field. In *Proceedings of 2013 3rd International Conference on Computer Science and Network Technology*, pages 1208–1211, 2013.

[44] D. Webb and J. van Den Berg. Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear differential contstraints. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2013.

[45] Q. Zhou, Y. Zhang, C. Rabbath, and D. Theilliol. Design of feedback linearization control and reconfigurable control allocation with application to a quadrotor uav. In *2010 Conference on Control and Fault-Tolerant Systems (SysTol)*, pages 371–376.