VISION BASED LOCALIZATION AND CONTROL OF UGVS USING A UAV

by
FARJAD ZAFAR

Submitted to
the Graduate School of Engineering and Natural Sciences
in partial fulfilment of
the requirements for the degree of
Master of Science

Sabancı University
December 2021

VISION BASED LOCALIZATION AND CONTROL OF UGVS USING A UAV

Approved by:

Prof. Dr. Mustafa Ünel .....
(Thesis Advisor)

Assoc. Prof. Kemalettin Erbatur .....

Assoc. Prof. Dr Hüseyin Üvet .....

Date of Approval: 14/12/2021

VISION BASED LOCALIZATION AND CONTROL OF UGVS USING A UAV

FARJAD ZAFAR

ME, Master's Thesis December 2021

Thesis Advisor: Prof. Dr. Mustafa Ünel

Keywords: State Estimation, Unmanned Aerial Vehicles, Unmanned Ground Vehicles, Visual Odometry, Localization

Abstract

Heterogeneous systems especially those that pair UAVs with UGVs are becoming increasingly popular. The advantages a multi-agent robot system brings are desirable for a multitude of tasks such as search and rescue, surveillance and reconnaissance. The mobility of UAVs by far makes them a better fit for perception and mapping tasks while the power and interaction capabilities of UGVs make them the desirable mode of operation for several other tasks. Thus, a system that looks to utilize them cooperatively has to solve the integral problem of localizing both robots in the same global map. Adding to this problem is the fact that a UAV-UGV pair is often desirable in scenarios where a common global sensor system such as GPS may not be available. In this thesis, a method to localize both vehicles in a single map is proposed by utilizing inertial sensors based estimates which are corrected through a single visual sensor onboard the UAV. We explore a method of generating a unified map which can be used to localize both the UGVs in the area as well as the UAV by levying the additional charge of conducting visual odometry and object detection on the UAV.

İHA KULLANARAK İKA'LARIN BILGISAYAR GÖRÜ TABANLI KONTROLÜ VE LOKALIZASYONU

FARJAD ZAFAR

ME, YÜKSEK LİSANS TEZİ, Aralık 2021

Tez Danışmanı: Prof. Dr. Mustafa Ünel

Anahtar Kelimeler: Durum Tahmini, İnsansız Hava Araçları, İnsansız Kara Araçları, Görsel Odometri, Yerelleştirme

ÖZET

Heterojen sistemler, özellikle İHA'ları İKA'larla eşleştiren sistemler giderek daha popüler hale geliyor. Çok etmenli bir robot sisteminin getirdiği avantajlar, gözetleme veya arama ve kurtarma gibi çok sayıda görev için arzu edilir. Çok etmenli bir robot sisteminin getirdiği avantajlar, gözetleme veya arama ve kurtarma gibi çok sayıda görev için arzu edilir. İHA'ların hareketliliği, onları algılama ve haritalama görevleri için daha uygun hale getirirken, İKA'ların güç ve etkileşim yetenekleri onları çoğu görev için arzu edilen çalışma modu haline getirir. Bu nedenle, onları işbirliği içinde kullanmayı düşünen bir sistem, her iki robotu da aynı küresel haritada yerelleştirme sorununu çözmek zorundadır. Bu soruna ek olarak, GPS gibi ortak bir küresel sensör sisteminin mevcut olmayabileceği senaryolarda genellikle bir İHA-İKA çiftinin istenmesi gerçeğidir. Bu tezde, İHA üzerindeki tek bir görsel sensör aracılığıyla düzeltilen ataletsel sensör tabanlı zayıf tahminler kullanılarak her iki aracı tek bir haritada konumlandırmak için bir yöntem önerilmiştir. İHA'da görsel odometri ve nesne algılama yürütmenin ek maliyetini alarak hem bölgedeki İKA'lar hem de İHA'yı lokalize etmek için kullanılabilecek birleşik bir harita oluşturma yöntemini araştırıyoruz.

ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest gratitude to my thesis advisor Prof. Dr. Mustafa Ünel, who with unwavering patience provided his earnest support and guidance. His advice and candor has allowed me to better myself as a researcher and the lessons I learned from him will undoubtedly stay with me throughout my career.

I would like to express my appreciation for Assoc. Prof. Kemalettin Erbatur and Assoc. Prof. Hüseyin Üvet for taking the time to read and provide reviews for my thesis and also being present on my jury committee.

I would also like to thank my graduate instructors who, through well constructed coursework and challenging projects enabled me to develop an appreciation for my subject and develop my research interests.

My friends and colleagues at the mechatronics laboratory in Sabanci including Usamah Yaseen, Naida Fetic and Selim Ahmet İz deserve a special mention for their support and assistance when I needed it most.

I am extremely grateful to my parents for their love, support, prayers and sacrifice that helped educate me and helped me prepare for my future. Last but not the least, I am privileged to mention my wife, Khunsha, who stood as a constant pillar of support and continuously encouraged me to better myself throughout my Masters.

To my family who despite the difficulties the last year presented, stood by me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The development of new sensors and processing units as well as the industrial success of stationary robots in their attempt to autonomously perceive and carry out tasks in their environment has generated tremendous interest in the field of mobile robotics. With abundant research being actively carried out in the areas of sensor fusion, precision navigation, precision positioning systems and obstacle avoidance over the last decade. Today, multiple solo mobile robot platforms such as Boston Dynamics Spot, have leveraged on-board multi-sensor systems to achieve a degree of autonomy in their tasks. However, single robot systems regardless of their robustness are often limited in their capabilities either due to design constraints such as locomotion mechanisms or by mission requirements such as in spatially separate tasks.

As a consequence, the idea of using heterogeneous systems wherein different kind of mobile robots cooperatively work together has gained popularity. Heterogeneous systems can consists of teams of two or more different types of vehicles such as Unmanned Aerial Vehicles (UAVs), Unmanned Ground Vehicles (UGVs) or Unmanned Surface Vehicles (USVs). The difference in shape, size, design and mode of locomotion between these robots allows a heterogeneous system to be employed in a broader spectrum of automated tasks. Perception, payload and computation tasks can be

easily distributed across the system to vehicles that best suit mission requirements. Evidently, the architectural scheme of co-dependence that a heterogeneous system introduces allows cooperating robots to accomplish complex tasks that would otherwise be impossible for a single powerful robot to accomplish. The fundamental theory of such a system encourages breaking down a difficult task into simpler subroutines and pass them to individual robots such that they interact to find ideal solutions. This allows cost effective and simpler robots to be used in lieu of a single costly robot with multiple capabilities.

A common combination utilised in heterogeneous systems is the UGV-UAV multi robot system. Such a system provides the agile and vantage point benefits from the UAV while keeping the payload and power capabilities of a UGV. This combination is of particular interest since the UAV can cover a larger spatial area and can provide information that a UGV might not be able to cover with the same on board sensors. This opens possibilities for better mapping and an additional layer of localization for the UGV using the UAV.

1.1 Motivation

UAV-UGV pairs are becoming increasingly popular in robotics. Recently, NASA launched the Ingenuity, a quad rotor UAV, alongside the Perseverance UGV rover to Mars. The Ingenuity has made the exploration on Mars efficient and safer for the rover by being able to provide additional mission data. Similar advantages are also sought in search and rescue missions where a UAV-UGV pair can definitively stand out compared to a single unmanned robot.

While advantageous, for most heterogeneous systems, it is imperative that the unmanned vehicles are able to localize themselves with respect to each other. This puts emphasis on the unmanned vehicles to be able to draw alignments between their maps and to continuously update their positions and estimate the difference

on the respective robots coordinate frame globally. Michael et al. [1] suggested using a common point for initializing all robots in a system to overcome this problem, while in [2] it is suggesting that using global positioning sensors can allow for good estimation of relative coordinate frames. A more recent and popular approach is to generate a reference map through one of the agents in the system while using an active sensor such as LIDAR or laser sensor to generate metrical map merges attached on the other robot [3].

The aforementioned while being increasingly viable according to their order of being stated, they each have their own drawbacks. The first method presents an ideal scenario which might not always be the case in the real world, the second method depends on sensors such as GPS, such sensors are closely dependant on the availability of satellites and their utility may be limited in certain scenarios and finally the last mentioned method is reliant on heavy and costly sensors.

As an alternative, rather than having to generate two different maps and merging them, this thesis suggests using a single map generated through the UAV and localize both vehicles through it. To this end it is suggested the UAV utilize a monocular camera to generate a map and localize the UGV through it by using computer vision algorithms.

1.2 Thesis Contribution

The contributions of this thesis are summarized as follows:

- A heterogeneous system where a UAV maps the environment and localizes UGV in the camera frame is presented.
- The models are simulated in a simulation environment where the physics are meticulously defined to exhibit the real world.

1.3 Thesis Outline

This thesis is organized into 6 chapters:

- Chapter 2 of the thesis is dedicated to conducting a literature survey regarding current UAV-UGV pairs and how they coordinate.
- Chapter 3 describes the mathematical modeling of both the UGV and UAV, with multiple models of the UGV discussed.
- Chapter 4 takes a look at the computer vision algorithms that will be used to carry out state estimation tasks.
- Chapter 5 explains how simulations are performed and provides a discussion about their results.
- Chapter 6 concludes the thesis with several remarks and indicates possible future directions.

Chapter 2

Background and Literature Survey

As previously stated heterogeneous robot systems consists of two or more than two different types of robots. This chapter presents the literature survey regarding the pair of UAVs and UGVs commonly used in research as well as the localization and mapping techniques utilized on such systems.

2.1 Unmanned Aerial Vehicles

An Unmanned Aerial Vehicle (UAV) belongs to a class of robots that operate above the surface of the earth. Generally, UAVs can be broadly divided into three categories with examples shown in Figure 2.1.

- Fixed-Wing: These types of UAVs adopt the age-old customary designs of aircrafts. Often used for military surveillance and weather forecast purposes. Fixed-Wing UAVs are primarily favoured for their high-altitude and high cruise speed capabilities.
- Rotary-Wing: This class of UAVs refers to crafts that utilize horizontal rotating aerofoils to generate lift. Helicopters as well as quadcopters belong to this class. The VTOL and hover capabilities that rotary-wing UAVs bring are of particular importance.

- Hybrid: This class of UAVs merges both the VTOL capabilities of rotary-wing as well as the cruise speed and fuel efficiency of a fixed wing UAV. Tilt-rotor UAVs are a prime example of the hybrid class.

Figure 2.1: Left:Rotary Wing DJI Quadcopter. Middle: Fixed Wing AA Albatross. Right: PD1 UKR Hybrid

.

Among the UAV classes, rotary-wings, particularly quadrotors are very popular in both solo-robot SLAM tasks as well as multi-robot tasks. This is partly due to their easy availability and the plethora of controls and modeling research that has been carried out on them [4]. However, arguably the most desirable quality for quadrotors is their versatility in being able to carry out missions with both close and far proximity from the ground and other unmanned agents in the area.

Multiple works have taken advantage of this versatility, such as in [5], where a UAV is used to conduct surveillance by taking the weak perspective approximation. Through this, the onboard camera of the UAV can be considered far away from the Earth such that the image plane is parallel to the ground plane, effectively allowing internal depth differences between objects on the ground to be disregarded [6]. Similarly, quadrotors have been used in a variety of different mission settings to detect obstacles either through a birds-eye view of the ground or to detect trajectory of obstacles in its paths to initiate obstacle avoidance maneuvers [7], [8].

Fixed-wing UAVs tend to be the more staple choice for tasks where a relatively large spatial area needs to be covered in a short amount of time. In recent works, fixed-wing UAVs equipped with cameras have been used to autonomously detect the

depth of snow in alpine conditions in order to help understand climate change over a period of time [9]. The work of Pfeifer et al. used a similar technique to detect and approximate the population of penguin colonies in Antarctica [10].

While quadrotors are currently the more popular choice for UAV-UGV pair systems, earlier research was focused on mobile target tracking using fixed-wing UAVs. In [11], target is separated from the background using a Laplacian operator based method and its velocity estimated by performing coordinate transformations and Kalman filtering.

2.2 Unmanned Ground Vehicles

Unmanned Ground Vehicles refers to all mobile robots that carry their tasks while retaining contact with the ground either autonomously or by passing information to a ground station. One of the first UGV dates back to 1904, it was called the RCA and was constructed by Leonardo Torres-Quevedo. The vehicle was tri-wheeled and was meant to be controlled through radio signals. Its simplistic tricycle model can be seen in Figure 2.2.

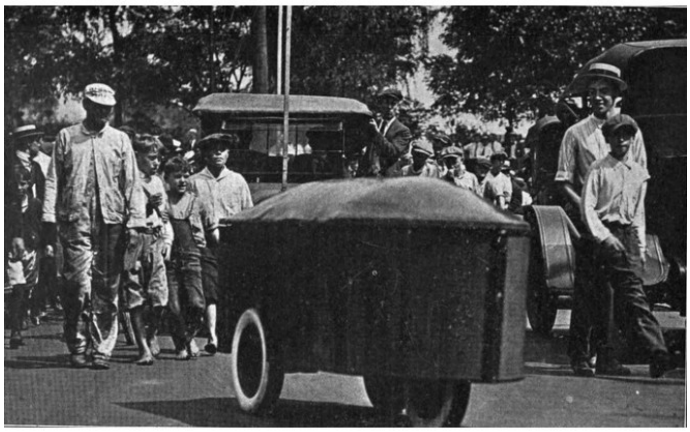


Figure 2.2: RCA revealed to the public

Today, multiple types of UGVs based on traditional vehicles as well as having been

inspired from biological organisms exist. We can loosely classify them as wheeled robots and legged robots. The term loosely is used since a very wide range of locomotion mechanisms exist within these classes, as well as with the current progress in soft-robotics an even larger array of UGVs may exist. But for clarity and with the perspective of this thesis, the later will not be considered as a category among UGVs.

2.2.1 Legged UGVs

Legged UGVs have long been a subject of interest due to their ability to tread rough terrain. The kinematics involved in legged robots give it the advantage of acting as a natural suspension allowing the motion of the main body to be fairly independent of how the legs interact with the ground. This is emphasized from the fact that one of the earlier legged UGVs developed in the 1980s was named the adaptive suspension vehicle [12].

Generally, there is no limit to the number or type of limbs a legged robot may have. Designs can vary from a single legged hopping robot [13], to complex sixteen legged large mechanical ones [14]. The primary problem in legged robots is the problem of keeping its balance and the ability of the robot to predict where it should land its feet even in cases of strong perturbations. A static system with external forces will be considered stable if its point of contact with the ground (assuming its an infinitesimally small point) is directly below its center of mass. If the points of contacts are increased e.g. in the case of a bi-pedalled robot that static stability is achieved when the center of mass is along the line between the two points. Additionally if more than three points of contact exist, than they will form the support polygon and as long as the center of mass remains within this polygon, balance is retained [15]. Figure 2.3 provides a visual representation of how center of mass and points of contacts are related for static stability.

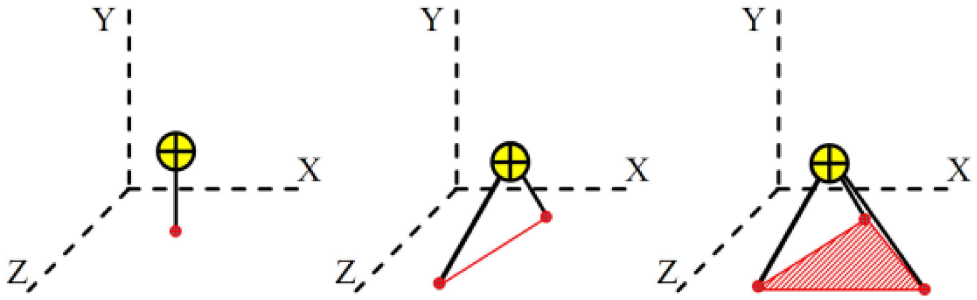


Figure 2.3: a: Stability for single point contact. b: Stability for dual point contact. c: Stability for three points of contact.

In [16], the standard approaches to modeling and controlling bi-pedalled robots are presented. And again in [17], a method of allowing bi-pedal robots to behave more closely to humans is presented. Quadruped and hexapod robots have received similar attention, with their design purposed more closely to work for disaster relief and surveillance mission [18].

2.2.2 Wheeled UGVs

The most typical form of a wheeled UGV is a standard differential drive robot. The differential drive robot generally consists of one or two castor wheels with two motor controlled wheels whose rotation rates can be separately altered. One of the most widely used differential drive robots for research are from the Turtle-bot family. The differential drive robot is preferred due to its simpler mathematical modeling and its accessible library and simulation environment on the robot operating platform ROS. New control algorithms and efficient path planning techniques such as in [19] are often carried out on this system.

Another widely researched UGV is the omni-wheeled UGV. These UGVs are equipped with wheels that have small discs around the circumference which are perpendicular to the turning direction. Due to this the wheel has the capability of easily slipping sideways. Omni-wheeled UGVs derive their research interest from their ability in making the drive system holonomic. Advanced control techniques and efficient ma-

neuverability are still of great interest in these UGVs. In [20] a fuzzy controller for improved response in robot soccer is proposed for omni-wheeled UGVs. Figure 2.4 shows an example for a typical arrangement in differential and omni-wheeled robots.

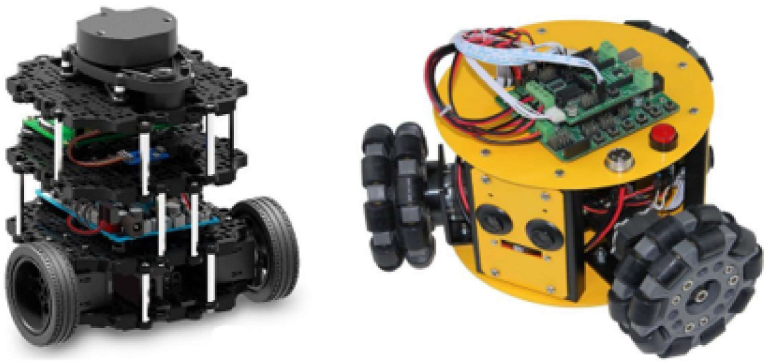


Figure 2.4: Left: TurtleBot Burger. Right: Typical Omniwheel Configuration

Among all wheeled UGVs however, the standard 4 wheeled configuration in terms of real life application is the most important. While new algorithms may be tested on differential drive UGVs ultimately, they are all designated to be used on the everyday car configuration. Today, a plethora of work is being carried out on developing a level 5 autonomous car, with new SLAM algorithms being tested on standardised benchmarks, such as the EuRoC or TUM dedicated primarily for this purpose [21]. For this UGV configuration and for the goal of attaining level 5 autonomy, a greater stress is laid on how such a system can accurately estimate its own state as well as the state of agents in its surrounding environment [22].

2.3 UAV-UGV Pairs

As mentioned previously, a UAV-UGV pair is a heterogeneous platform that has a wide array of advantages over a using either of the two in a singular platform. The work carried out in literature is generally purposed towards methods of coordinating between the pair in order to carry out simple to complex tasks.

The strategies undertaken for multi-agent tasks involve giving top level decision taking authority to one agent or to levy task priority among all available agents in the system. The type of system being used depends on the level of hierarchy given to the number of agents present in the system. Depending on this coordination strategies can be divided into Distributed systems, Centralized Systems and Decentralized Systems. We take a brief look at how these methods are used in literature in the specific case of UAV-UGV pairs, a graphical description of these systems is provided in Figure 2.5.

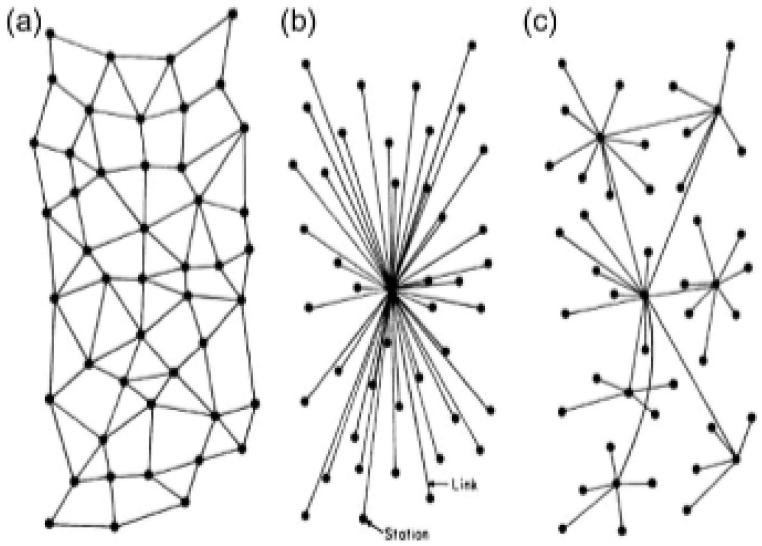


Figure 2.5: a: Distributed System. b: Centralized System. c: Decentralized System

2.3.1 Distributed Coordination Systems

In the distributed coordination architecture, no single robot is in charge of making unilateral decisions for other agents in the system. Instead each robot in the system is given an equal degree of autonomy and decision making preference [23]. Each robot in the system is equipped with its own set of sensors and computational processors and is responsible for generating its own map of the environment. The robots do however, exchange information and holistically develop the entire map model of the covered spatial area by sharing their own local maps. While distributed sensors

can be effective in developing a more accurate representation of the world, they are not very effective in dynamic environments.

A distribution coordination system is used in [24] to generate multi view map by using multiple agents in the system. Each robot is responsible for carrying out its own SLAM task in it local environment while simultaneously updating the global map by sharing map information. The method presented here maintains a topological pose graph which is stored in nodes and edges. Since only reduced graph structure is shared, maps from multiple robots can be merged to create a large global map. The maps are optimally merged by using visual features detected by individual robots in the distributed system.

In another work, the authors have used a UAV and UGV pair to collaboratively localise and generate an elevation map based on 2 different sensors [25]. The UAV is equipped with a monocular camera and generates an elevation map using visual features, while the UGV is equipped with a rotating laser which it uses to generate its own elevation map. The robots continuously update their elevation maps and share this data with each other. This allows the ground robot to use the aerial view point as a reference map to better estimate its own states while the UAV can use this information to localize the UGV in its own local map. A great advantage of this technique is that the method is independent of having to calculate the respective viewpoint of each vehicle.

2.3.2 Centralized Coordination Systems

The main concept behind centralized coordination is to assign decision making authority to one agent in the system. This agent makes use of its on board sensors to perceive the environment and to make control decisions for itself as well as all the other agents. It is essentially, the brain of the entire architecture that alone ensures how tasks are to be carried out. An euphemism popularly used for this approach in

literature is the Leader-Follower control formation [26], [27].

Since the centralized approach simply extends the modality of agents in single robot systems. This makes the architecture computationally simpler. However, since the entire system is dependant on the leader to receive information and perform loop closure for follower robot responsibility, it becomes liable to communication over head and transmission loses. The entire system is liable to become unstable if the leader malfunctions unless an alternative robot exists [28]. In [29], a centralized architecture is used to conduct surveillance using a UAV-UGV pair.

In [30] the authors present a fully autonomous collaboration between a UAV and UGV in a mock-up disaster scenario. The scenario builds on the fact that the ground robot is unable to find the ideal path through obstacles, and hence the UAV carries out a prior mission to map the area of interest first, and then through the aid of fiducial markers localizes the UGV in the global map and guides it through the optimized way points.

Garzon et. al. in his work also uses a centralized approach to map obstacles and navigate a UGV using visual feedback received from the UGV [31]. The UGV uses visual information from the camera to detect obstacles in its path and utilizes its own inherent positioning system to be able to localize itself with respect to the objects in its path.

2.3.3 Decentralized Coordination System

This architecture presents a hybrid approach between centralized and distributed coordination systems. In this paradigm the control process is achieved by giving one or more local centric robots within the system to carry out decision making for the agents within their immediate vicinity. This causes groups of clusters to be formed

throughout the system where each group is responsible for carrying out sub tasks individually in a centralized manner [32]. This form of coordination is robust compared to distributed architecture and allows the system to perform tasks efficiently through global goals and plans. Decentralized systems are often the more popular approach in heterogeneous systems.

2.4 Simulation Environment

CARLA is an open source Simulation software developed by Toyota in the Unreal Engine environment for the study and development of machine algorithms for autonomous driving. Using the Unreal Engine physics, it allows a ground vehicle to be simulated within an open world environment where data can be extracted in different forms from simulated sensors. Some sensors featured by CARLA are LIDAR, Depth Camera, Collision Sensor and most importantly for this thesis, the RGB camera. CARLA also comes equipped with a complex traffic manager which simulates pedestrian and vehicle NPC behavior. In addition to the intrinsic capabilities of the simulator that embeds the control logic, physics and graphical rendering of the environment and characters within it, it includes a Python API module which allows user generated code and existing libraries such as pytorch or openCV to interact with the simulation [33]. Figure 2.6 provides a basic idea of how the inherent architecture is separate from the python API.

Most aspects of the simulation can be accessed from the API. With scripts it is possible to generate, retrieve and process raw data coming from and into CARLA sensors. This data can be additionally processed for control actions. We can divide the python scripts into two logical parts.

- Configuration: Before any valid work can be done on CARLA, a connection with the CARLA server needs to be established and the objects of interest

including sensors, vehicles or else need to be created, spawned or destroyed.

- Communication: This part of the simulation is dedicated to carrying out user generated actions. This includes receiving data from the sensors, processing them and actuating the vehicle through user defined control logic.

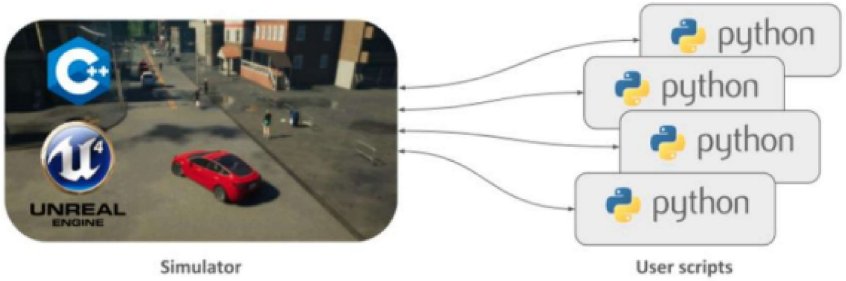


Figure 2.6: Carla Python API

Chapter 3

Mathematical Modeling of UAV and UGV

Mathematical models are necessary to describe how a system will behave in the simulation environment. In this chapter the mathematical models for the unmanned vehicles to be used in our simulation environment are described. These mathematical models will be used to navigate the UAV through the virtual environment and to consequently control the UGV from state observations made by the camera and internal sensors of the UGV.

3.1 Quadrotor Modeling

The quadrotor configuration of a UAV consists of four rotors arranged in a cross configuration. The angular rotation of each rotor can be independently controlled to produce different thrusts. How the angular rates of these rotors are changed primarily govern how the quadrotor will react. The pitch and roll moments can be controlled by varying the speeds of rotors arranged in a cross, while the acceleration in altitude can be controlled using the sum of thrust for all four motors. Finally, yaw control of the quadrotor is obtained from the net torque of all four motors. These relations are depicted in Figure 3.1.

To this end we will first discuss the axes system used to analyze the quadrotor dynamics followed by the transformation from one axes system to another. At the end we will formulate the quadrotor dynamics and present equations of motion for 6-dof.

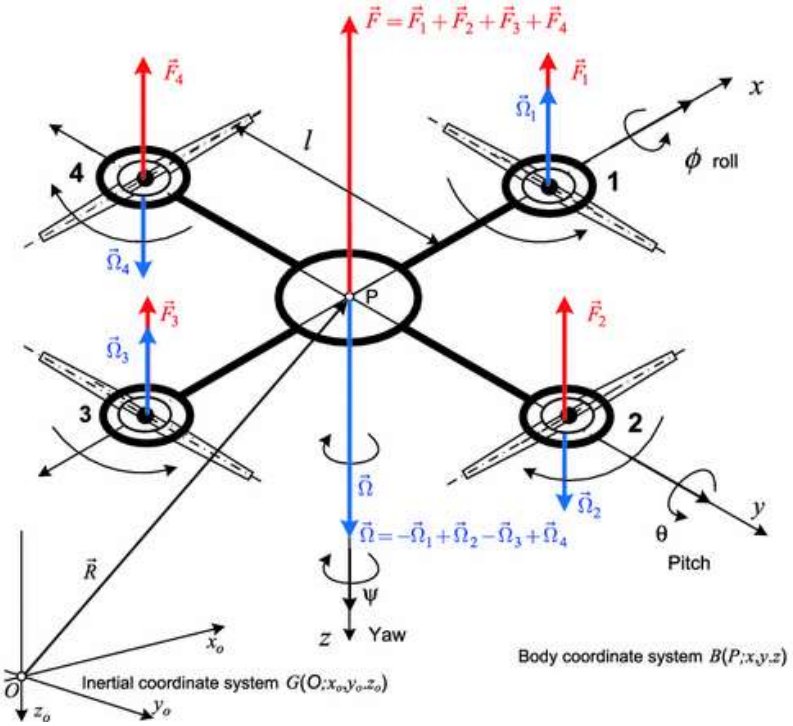


Figure 3.1: Forces and Angular Rates of a Quadrotor.

3.1.1 Assumptions

A real system involves numerous variables that when considered increasingly complicates the modeling manifold. This complexity increases computation time and does not prove too much in the way of accuracy for the implementation in this thesis. Hence, we take reasonable assumptions that do not inhibit the model from representing the actual outcome by too large a margin.

- Rigid Structure: This implies that no body distortions or blade flapping take place. Additionally, the relative distance between any two points on the propellers will remain constant [34].
- Symmetric Structure: Based on the assumption the mass of the quadrotor including any on board sensors is considered to be symmetrically distributed

along all three axis. This allows us to take elements not on the main diagonal of inertia to be taken as zero.

- Flat, non-rotating Earth: This assumption allows us to ignore the orbital motion of the Earth.
- Negligible Ground Effect: In our simulations the quadrotor is expected to be at a certain height from the ground. Take-off phase of the mission can largely be ignored since the quadrotor is only expected to start estimating the states of the UGVs once it is on its flight path.
- Constant Mass: The mass of the quadrotor will not change throughout operation.

3.1.2 Axes Systems

The following axes systems will be used for the dynamic modeling.

Inertial Axes System

Specifying an inertial axis is important since Newton's laws of motion are only applicable when acceleration is measured with respect to an Earth-fixed frame. Hence, to formulate the dynamics problem we take a non-rotating reference frame placed at the center of the Earth as the inertial frame of reference. This frame is placed in the NED configuration as listed below.

- Positive x-axis is directed towards the North.
- Positive y-axis is directed towards the East.
- Positive z-axis is directed downwards towards the center of the Earth.

Body Axes System

The center of mass of the quadrotor is taken as the origin for the body axis system. The body axes system allows the moments and products of inertia as well as the

forces and moments acting on the quadrotor to be specified easily. This system is taken as orthogonal right-handed.

- x-axis passes through rotor 1.
- y-axis passes through rotor 2.
- z-axis represents downwards direction.

Transformations between Reference Frames

As the body-fixed frame is a rotating frame of reference which moves along with the quadrotor, Newton's laws can not be directly applied to it. Hence, there is a need to transform the variables calculated in the body frame to the inertial frame. There are three general methods available to do so.

- Euler Angles
- Quaternions
- Direct Cosine Matrices

Of these methods the most intuitive and computationally inexpensive technique is to use the Euler Angles. And since the quadrotor is not expected to carry out any aggressively maneuvers, the singularity problem associated with this method can be largely ignored. The rotation matrices associated with the z, y and x axis respectively are given below.

$$R_z(\psi) = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \tag{3.2}$$

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \tag{3.3}$$

The sequence of rotations of these matrices are defined in Equation 3.4.

$$R_I^B = R_2^B(\phi) R_1^2(\theta) R_I^1(\psi) \tag{3.4}$$

$$R_I^B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The final transformation is represented Equation 3.5.

$$R_I^B = \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \cos\theta\sin\phi \\ \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi & \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi & \cos\theta\cos\phi \end{bmatrix} \tag{3.5}$$

Since the individual transformation matrices are orthogonal, the final matrix is also orthogonal.

$$[R_I^B]^{-1} = [R_I^B]^T$$

Now we can easily transform a vector from the body frame to the inertial frame.

$$X_I = R_I^B X_B$$

These Euler angle rotations are also graphically represented in Figure 3.2.

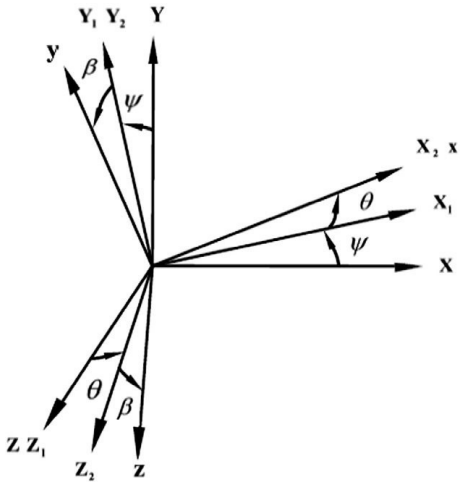


Figure 3.2: Euler Angle Rotations [35]

3.1.3 Kinematic Relations

Modeling relations that describe the motion of an object without accounting for the forces acting on it are known as the kinematic relations. These relations are used to describe how an object translates and rotates.

Using the sequence of transformations stated previously angular rates (p,q,r) can be expressed in terms of Euler angle rates in the body frame.

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + R_x(\phi) \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + R_x(\phi)R_y(\theta) \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \tag{3.6}$$

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{3.7}$$

And in terms of Euler angle rates we represent this with the relation given below.

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sec\theta\sin\phi & \sec\theta\cos\phi \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{3.8}$$

By integrating these equations the attitude of the quadrotor can be obtained in the inertial frame of reference.

Similarly, the same matrix declared earlier can be used to find the position and velocity of the quadrotor in the earth frame. We mathematially describe these equations below.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = R_B^I \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{3.9}$$

$$\dot{x} = (\cos\psi\cos\theta)u + (\cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi)v + (\cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi)w \tag{3.10}$$

$$\dot{y} = (\sin\psi\cos\theta)u + (\sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi)v + (\sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi)w \tag{3.11}$$

$$\dot{z} = -(\sin\theta)u + (\cos\theta\sin\phi)v + (\cos\theta\sin\phi)w \tag{3.12}$$

Through integration, the position of the quadrotor in the inertial frame can be calculated.

3.1.4 Dynamic Relations

Now that we have defined the kinematic relations for the quadrotor, we now move to the mathematical relations that govern how forces and moments cause motion. These relations also known as the dynamic relations can be built for the quadrotor

starting from Newton's second law of motion.

$$F = \frac{d}{dt}(mV) \mid_I$$

$$\begin{bmatrix} F_X \\ F_Y \\ F_Z \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} -mg\sin\theta \\ mg\cos\theta\sin\phi \\ mg\sin\theta\cos\phi \end{bmatrix} = \begin{bmatrix} m(\dot{u} + qw - rv) \\ m(\dot{v} + ru - pw) \\ m(\dot{w} + pv - qu) \end{bmatrix} \tag{3.13}$$

Here the term d/dt is not carried since mass is assumed to be constant. These equations can now be solved for acceleration.

$$\dot{u} = rv - qw - g\sin\theta + \frac{1}{m}F_{xd} \tag{3.14}$$

$$\dot{v} = pw - ru + g\cos\theta\sin\phi + \frac{1}{m}F_{yd} \tag{3.15}$$

$$\dot{w} = qu - pv + g\cos\theta\cos\phi + \frac{1}{m}F_{zd} \tag{3.16}$$

The force disturbance terms F_{xd}, F_{yd} and F_{zd} act on their respective axis as drag.

$$F_{xd} = -\frac{1}{2}C_D A_D \rho u.|u| \tag{3.17}$$

$$F_{yd} = -\frac{1}{2}C_D A_D \rho v.|v| \tag{3.18}$$

$$F_{zd} = -\frac{1}{2}C_D A_D \rho w.|w| \tag{3.19}$$

where A_D and C_D are aerodynamic area and drag coefficient respectively.

As moments are also integral for defining the dynamic relations of the quadrotor. Similar to the force equation we can use the moment equation to describe the model.

$$M = \frac{d}{dt}(H) \mid_I$$

$$\begin{bmatrix} M_X \\ M_Y \\ M_Z \end{bmatrix} = \begin{bmatrix} L \\ M \\ N \end{bmatrix} = \begin{bmatrix} I_{xx}\dot{p} + (I_{zz} - I_{yy})qr \\ I_{yy}\dot{q} + (I_{xx} - I_{zz})rp \\ I_{zz}\dot{r} + (I_{yy} - I_{xx})pq \end{bmatrix} \tag{3.20}$$

As per the symmetric structure assumption, cross product terms are taken as zero and not represented in the equations above. Finally, the moments can be solved for in the inertial frame as,

$$\dot{p} = \frac{1}{I_{xx}}[(I_{xx} - I_{zz})qr + M_x + M_{xg}] \tag{3.21}$$

$$\dot{q} = \frac{1}{I_{yy}}[(I_{zz} - I_{xx})rp + M_y + M_{yg}] \tag{3.22}$$

$$\dot{r} = \frac{1}{I_{zz}}[(I_{zz} - I_{yy})qp + M_z + M_{zg}] \tag{3.23}$$

the moment disturbance terms M_{xg}, M_{yg}, and M_{zg}, in the aforementioned equations occur due to the gyro effect, which in quadrotors generally occurs when the rotors are not rotating at the same speed. We model this with the following equations.

$$\begin{bmatrix} M_{xg} \\ M_{yg} \\ M_{zg} \end{bmatrix} = J_r \begin{bmatrix} q\Omega_r \\ -p\Omega_r \\ \dot{\Omega}_r \end{bmatrix} \tag{3.24}$$

where, J_r is the inertia of the rotors and Ω_r is the residual rotational rate of rotors.

$$\Omega_r = \Omega_1 + \Omega_2 - \Omega_3 + \Omega_4 \tag{3.25}$$

The complete non-linear model of the quadrtor consists of 12 states which is represented by the vector equation $\dot{x}$.

$$\dot{x} = f(X, U) \tag{3.26}$$

Where X represents all the states of the system.

$$X = \begin{bmatrix} (\cos\psi\cos\theta)u + (\cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi)v + (\cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi)w \\ (\sin\psi\cos\theta)u + (\sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi)v + (\sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi)w \\ -(\sin\theta)u + (\cos\theta\sin\phi)v + (\cos\theta\sin\phi)w \\ rv - qw - g\sin\theta + \frac{1}{m}F_{xd} \\ pw - ru + g\cos\theta\sin\phi + \frac{1}{m}F_{yd} \\ qu - pv + g\cos\theta\cos\phi + \frac{1}{m}F_{zd} \\ p + \tan\theta(q\sin\phi + r\cos\phi) \\ q\cos\phi - r\sin\phi \\ (q\sin\phi + r\cos\phi)\sec\theta \\ \frac{1}{I_{xx}}[(I_{xx} - I_{zz})qr + M_x + M_{xg}] \\ \frac{1}{I_{yy}}[(I_{zz} - I_{xx})rp + M_y + M_{yg}] \\ \frac{1}{I_{zz}}[(I_{zz} - I_{yy})qp + M_z + M_{zg}] \end{bmatrix} \tag{3.27}$$

The input vector U represents the four inputs passed to the model to control the rotorcraft represented below.

$$U = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 \end{bmatrix}^T$$

$$U = \begin{bmatrix} b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2 \\ bl(-\Omega_2^2 + \Omega_4^2) \\ bl(\Omega_1^2 - \Omega_3^2) \\ d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2 \end{bmatrix} \tag{3.28}$$

Here u_1 represents total vertical thrust, u_2 represents pitching moment, u_3 represents rolling moment and u_4 the yawing moment. Constants b, d and l represent thrust factor, drag factor and length of moment arm respectively.

3.2 Ground Vehicle Modeling

Now that one of our agents to be simulated has been defined. We move towards describing the mathematical model of the ground vehicle. We take a 4 wheel vehicle and move to define how it is expected to behave. First we will define a simple kinematic mathematical model, followed by the dynamic model for the vehicle.

3.2.1 Kinematic Model

As described previously, kinematic models are used to describe motion by using the geometric constraints imposed on the system.For the 4-wheeled car we take the well known kinematic bicycle model.

In the bicycle model of a 4 wheeled vehicle the front left and right wheels are taken as a single unit, the same assumption is also taken for the two rear wheels. Both wheels are donated by δ_f and δ_r, such that it is assumed that both the front unified wheels as well as the rear unified wheels are steerable. If steer ability for only one set of wheels is required, the other can be set as zero. The center of gravity of the vehicle is located between the line passing through the front wheel and the rear wheel. For a better representation the following Figure 3.3 is used.

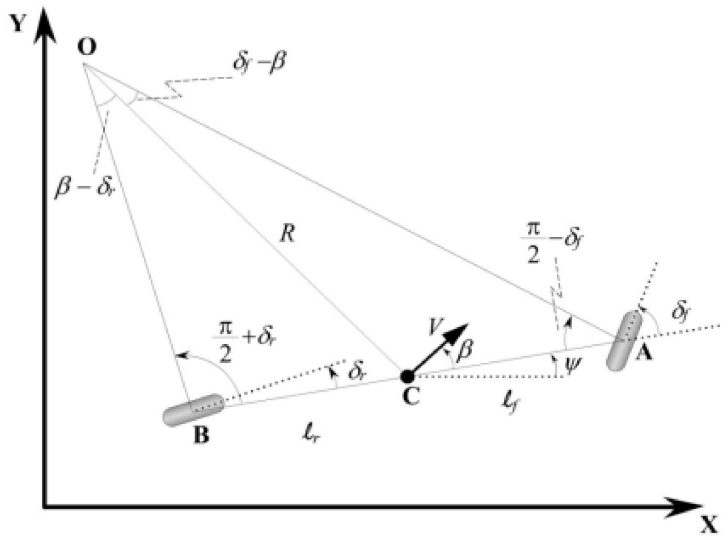


Figure 3.3: Kinematics of lateral vehicle motion [36]

A and B are the front and rear wheels respectively. C represents the center of gravity, while the distance of the front wheel and rear wheel from the C.G are given as l_f and l_r respectively. As a simplified assumption, in the kinematic model the vehicle is assumed to be in planar motion hence, x,y and ψ are required to describe the motion of the vehicle. Where x and y are the inertial coordinates of the center of gravity of the vehicle and ψ represents the orientation. The vector V at the c.g represents the velocity of the vehicle. A difference between the angle of the velocity at C and the heading of the vehicle may exist. This difference is denoted by β and is known as the side slip angle.

The point O in the Figure 3.3 above represents the instantaneous rolling center for the vehicle. This point is defined by the intersection between the imaginary lines drawn perpendicularly from the center of both wheels. The radius of the vehicles path is now defined by the how far the point O exists from the the c.g. By creating these imaginary lines we are able to obtain two similar triangles, OCA and OCB. By solving for them we can arrive at the following equation.

$$(\tan(\delta_f) - \tan(\delta_r)) \cos\beta = \frac{l_f + l_r}{R} \tag{3.29}$$

An integral assumption in the kinematic bicycle model is that the vehicle has a slow speed such that the rate of change of orientation of the vehicle becomes equal to the angular velocity of the vehicle. We can describe this using the relation,

$$\dot{\psi} = \frac{V}{R}$$

And by substitution we further obtain Equation 3.30.

$$\dot{\psi} = \frac{V\cos(\beta)}{l_f + l_r}(\tan(\delta_f) - tan(\delta_r)) \tag{3.30}$$

Now we can write the overall equations of motions.

$$\dot{x} = V\cos(\psi + \beta) \tag{3.31}$$

$$\dot{y} = V\sin(\psi + \beta) \tag{3.32}$$

$$\dot{\psi} = \frac{V\cos(\beta)}{l_f + l_r}(\tan(\delta_f) - tan(\delta_r)) \tag{3.33}$$

The slip angle can be similarly measure by carrying out the appropriate substitutions.

$$\beta = \tan^{-1}\left[\frac{l_f\tan\delta_r + l_r\tan\delta_f}{l_f + l_r}\right] \tag{3.34}$$

The Kinematic Bicycle model suffers from its base assumption that the vehicle is moving a at a very slow speed. However, in most scenarios the this assumption will be violated, hence we move to develop the dynamic model of the vehicle to overcome this short coming.

3.2.2 Longitudinal Dynamic Modeling

For describing the longitudinal dynamics of the car we take a vehicle placed on an inclined plane such as in Figure 3.4 and observe all the external forces acting on it.

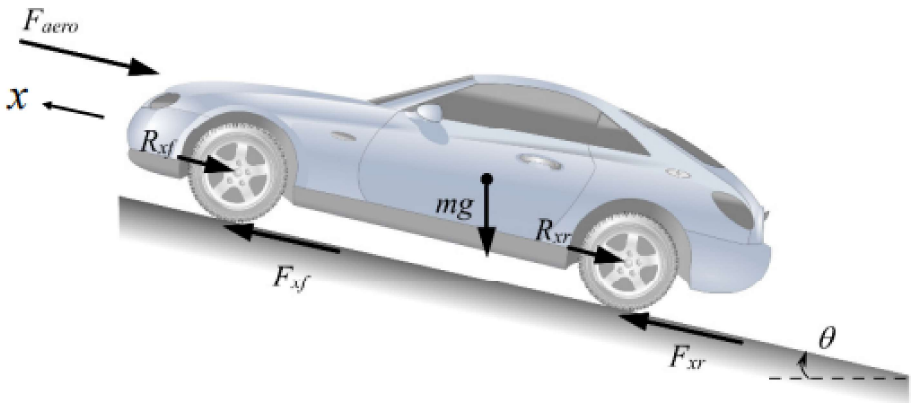


Figure 3.4: Forces on a body in an inclined plane [37]

Table 3.1: Parameters acting on an inclined vehicle

Parameter	**Description**
F_{xf}	Longitudinal tire force at front tires
F_{xr}	Longitudinal tire force at rear tires
F_{aero}	Equivalent longitudinal aerodynamic drag force
m	Mass of vehicle
g	Gravitational acceleration
θ	Angle of inclination
R_{xf}	Rolling resistance at front tires
R_{xr}	Rolling resistance at rear tires

Using the tire forces to describe the thrust being obtained from the power train and by balancing it with the resisting forces along the longitudinal axis of the vehicle we can obtain the force relations.

$$m\ddot{x} = F_{xf} + F_{xr} - F_{aero} - R_{xf} - R_{xr} - mg\sin(\theta) \tag{3.35}$$

Now we move towards better defining the force terms on the right hand side of the equation.

The aerodynamic drag force is the resistant force exhibited by the air. It depends on both the vehicle speed and the wind speed and for vehicles can be given with the relation in Equation 3.36.

$$F_{aero} = \frac{1}{2}\rho C_d A_F (V_x + V_{wind})^2 \tag{3.36}$$

In the aforementioned equation ρ refers to the density of air, A_F is the normal frontal area of the vehicle that is facing the direction of travel and C_d is the coefficient of aerodynamic drag. The wind term can be positive for a head wind or negative in case of a tail wind. [36] showed that frontal area is about 80 percent of the vehicle area calculated from width and height, they also showed that for cars in the mass range of 800-2000 kg can be given with the following relation.

$$A_F = 1.6 + 0.00056(m - 765) \tag{3.37}$$

The aerodynamic drag coefficient can be carried out using a simple coast down test [38]. Assuming that the inclination angle is zero, the throttle is small enough to be considered zero and wind speed is negligible next set of operations on the dynamics equation.

$$-m\frac{dV_x}{dt} = \frac{1}{2}\rho V_x^2 A_F C_d + R_x$$

By integrating for an initial longitudinal velocity V_0,

$$t = [\frac{2m^2}{pA_F C_d R_x}]^{\frac{1}{2}-1}[V_0(\frac{\rho A_F C_d}{2R_x})^{\frac{1}{2}}] - \tan^{-1}[V_x(\frac{\rho A_F C_d}{2R_x})^{\frac{1}{2}}] \tag{3.38}$$

if we take t=T as total time and add the non-dimesiinalizing parameter,

$$\beta = V_0(\frac{\rho A_F C_d}{2R_x})^{\frac{1}{2}} \tag{3.39}$$

$$\frac{V_x}{V_0} = \frac{1}{\beta}tan[(1 - \frac{t}{T}\tan^{-1}(\beta)] \tag{3.40}$$

$\frac{V_x}{V_0}$ represent a single family of curves that cane use to obtain β for a vehicle at any particular time. We can use these values to then calculate the drag coefficient and the rolling resistance.

$$C_d = \frac{2m\beta \tan^{-1}(\beta)}{V_0 T \rho A_F} \tag{3.41}$$

$$R_x = \frac{V_0 m \tan^{-1}(\beta)}{\beta T} \tag{3.42}$$

Next we move to better define the longitudinal tire forces in the front and rear wheels. These forces depend on a variety of factors including,

- Normal load on tire
- Friction coefficient
- Slip ratio

Longitudinal slip is the difference between the actual longitudinal velocity of the vehicle and the rotational velocity. The slip ratio for acceleration and breaking are respectively given as,

$$\sigma_x = \frac{r_{eff}\omega_w - V_x}{r_{eff}\omega_w} \tag{3.43}$$

$$\sigma_x = \frac{r_{eff}\omega_w - V_x}{V_x} \tag{3.44}$$

if the friction coefficient of the tire and road interface is assumed as one while the normal force to the tire is taken as constant a function of slip ratio to tire force is shown in Figure 3.5.

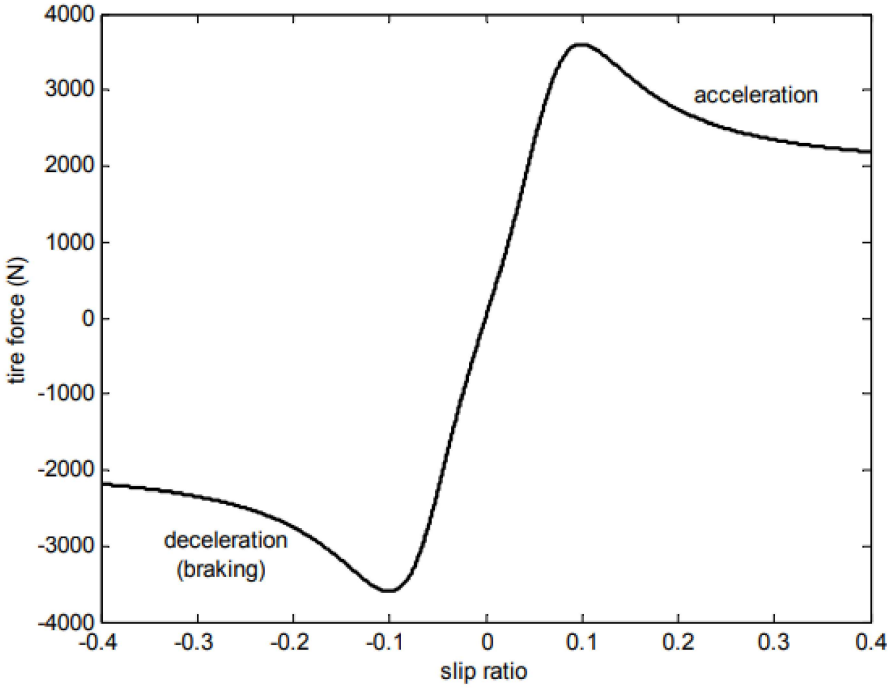


Figure 3.5: Tire Force as a function of slip

The linear region describes how the slip ratio remains during normal driving conditions. Taking this as a proportional relationship the tire forces can now be modeled as given in Equations 3.45 and 3.46.

$$F_{xf} = C_{\sigma f}\sigma_{xf} \tag{3.45}$$

$$F_{xr} = C_{\sigma r}\sigma_{xr} \tag{3.46}$$

The rolling resistance of the tires are roughy modeled as being proportional to the normal force on the tires as,

$$R_{xf} + R_{xr} = f(F_{zf} + F_{z_r}) \tag{3.47}$$

f here represents the rolling resistance coefficient which for typical passenger cars is 0.015 [36].

Apart from weight, the normal load on tires is also affected by additional parame-

ters such as the longitudinal acceleration, fore-aft location of c.g., the aerodynamic drag forces and the terrain. We model these by taking moments about the point of contact of the tires using Figure 3.6 as reference.

$$F_{z_f} = \frac{-F_{aero}h_{aero} - m\ddot{x}h - mgh\sin(\theta) + mgl_r\cos(\theta)}{l_f + l_r} \tag{3.48}$$

$$F_{z_r} = \frac{F_{aero}h_{aero} + m\ddot{x}h + mgh\sin(\theta) + mgl_r\cos(\theta)}{l_f + l_r} \tag{3.49}$$

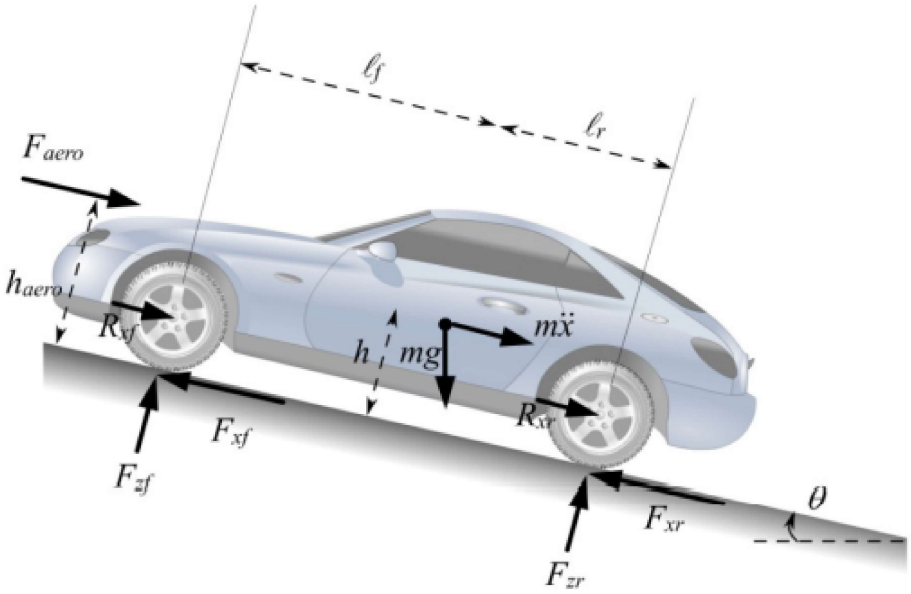


Figure 3.6: Normal Tire Loads Calculation

Table 3.2: Parameters for normal tire loads

Parameter	**Description**
h	Height of c.g.
h_{aero}	Height of aerodynamic center
l_f	Distance between front axle and c.g.
l_r	Distance between rear axle and c.g.
r_{eff}	Effective radius of tires

Driveline Dynamics

We defined how the longitudinal motion of the vehicle is dependant on the longitudinal forces of the tires. These tire forces are greatly dependant on the difference

between wheel rotational velocity and the vehicle longitudinal velocity. The rotational velocity of the wheel itself is greatly dependant on the driveline dynamics of the vehicle, which makes it an integral part of the model. A standard power train and its flow is defined in Figures 3.7 and 3.8.

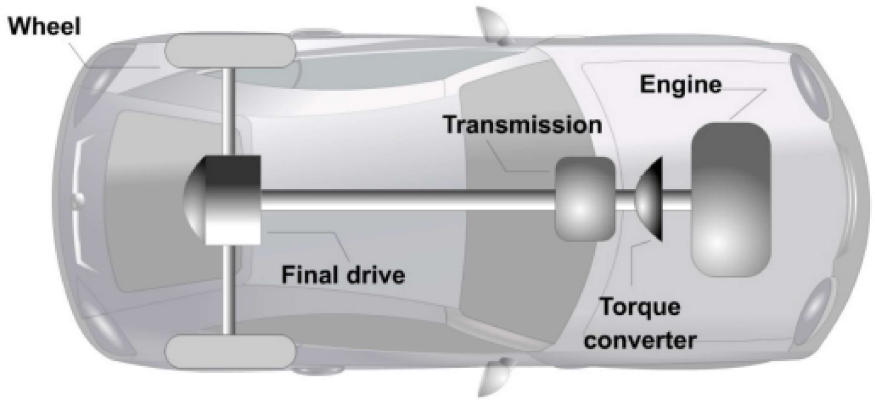


Figure 3.7: Typical Power Train

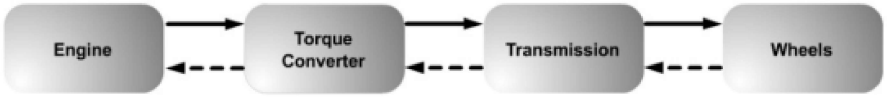


Figure 3.8: Power train flow (solid arrows represent power; dotted arrows represent load)

Due to the direct connection between wheel and engine when in gear, we can model this behaviour with a kinematic restraint such that rotational coupling exists at all time [39].

$$\omega_w = GR\omega_t = GR\omega_e \tag{3.50}$$

where ω_w is wheel angular speed, ω_t is turbine angular speed, ω_e is engine angular speed and finally GR is the gear ratio. We can also represent the longitudinal velocity in terms of angular wheel velocity.

$$\dot{x} = r_{eff}\omega_w \tag{3.51}$$

$$\ddot{x} = r_{eff} GR \dot{\omega_w} \tag{3.52}$$

The wheel represent the intersection between torques coming from the power train and the torques from external forces, we use these to model the dynamic behavior.

$$I_w \dot{\omega_w} = T_{wheel} - r_{eff} F_x T_{wheel} = I_w \dot{\omega_w} + r_{eff} F_x \tag{3.53}$$

$$T_{wheel} = I_w \dot{\omega_w} + r_{eff} F_x \tag{3.54}$$

Since we already have defined the equation for force exhibited by the tyres, we can solve this differential equation for wheel torque. Now the turbine torque can be defined as the torque which comes from the torque converter that couples the engine to the transmission. This can be modeled with the following relations.

$$I_t \dot{\omega_t} = T_t - (GR) T_{wheel} \tag{3.55}$$

$$I_t \dot{\omega_t} = T_t - (GR) T_{I_w \dot{\omega_w} + r_{eff} F_x} \tag{3.56}$$

Next is the torque converter which is perhaps the most complicated part of the drivetrain, the torque converter can undergo coupling and decoupling with the engine due to the multiple impellers and fluid flowing through it. For our purposes however, we consider the converter to be coupled at all times which allows us to use the relation in Equation 3.57.

$$\omega_t = \omega_e \tag{3.57}$$

$$T_t = (I_t + I_w G T^2) \dot{\omega_e} + GR r_{eff} F_x \tag{3.58}$$

Finally we add the engine dynamics where the engine inertia term is equal to the torque produced by the engine from the combustion process minus the torque from the torque converter.

$$I_e\dot{\omega}_e = T_{Engine} - T_t \tag{3.59}$$

$$I_e\dot{\omega}_e = T_{Engine} - (I_t + I_w GT^2)\dot{\omega}_e - GRr_{eff}F_x \tag{3.60}$$

From the dynamic modeling we have the following list of equations. Starting with the primary vehicle dynamic equation.

$$m\ddot{x} = F_{xf} + F_{xr} - F_{aero} - R_{xf} - R_{xr} - mg\sin(\theta) \tag{3.35}$$

For front longitudinal tire forces,

$$F_{xf} = C_{\sigma f}\sigma_{xf} \tag{3.45}$$

$$\sigma_{xf} = \frac{r_{eff}\omega_{wf} - \dot{x}}{r_{eff}\omega_{wf}} \tag{3.43}$$

$$\sigma_{xf} = \frac{r_{eff}\omega_{wf} - \dot{x}}{\dot{x}} \tag{3.44}$$

For rear longitudinal tire forces,

$$F_{xr} = C_{\sigma r}\sigma_{xr} \tag{3.46}$$

$$\sigma_{xr} = \frac{r_{eff}\omega_{wr} - \dot{x}}{r_{eff}\omega_{wr}} \tag{3.43}$$

$$\sigma_{xr} = \frac{r_{eff}\omega_{wr} - \dot{x}}{\dot{x}} \tag{3.44}$$

For rolling resistances,

$$R_{xf} + R_{xr} = f(F_{zf} + F_{z_r}) \tag{3.47}$$

$$F_{z_f} = \frac{-F_{aero}h_{aero} - m\ddot{x}h - mgh\sin(\theta) + mgl_r\cos(\theta)}{l_f + l_r} \tag{3.48}$$

$$F_{z_r} = \frac{F_{aero}h_{aero} + m\ddot{x}h + mgh\sin(\theta) + mgl_r\cos(\theta)}{l_f + l_r} \tag{3.49}$$

For Aerodynamic Drag force,

$$F_{aero} = \frac{1}{2}\rho C_d A_F (V_x + V_{wind})^2 \tag{3.36}$$

Finally the combined engine dynamic model,

$$(I_e + I_t + IwGR^2 + m(GR^2)r_{eff}^2)\dot{\omega}_e = T_{engine} - (GR)(r_{eff}F_x) \tag{3.61}$$

3.2.3 Lateral Dynamics

Moving with the same assumption as in the kinematic model, we lump the two front wheels and the two rear wheels together at the center of the car to generate a bicycle model again. The model is considered to be planar and has two degrees of freedom i.e. the vehicles lateral position and the vehicles yaw angle such as in Figure 3.9.

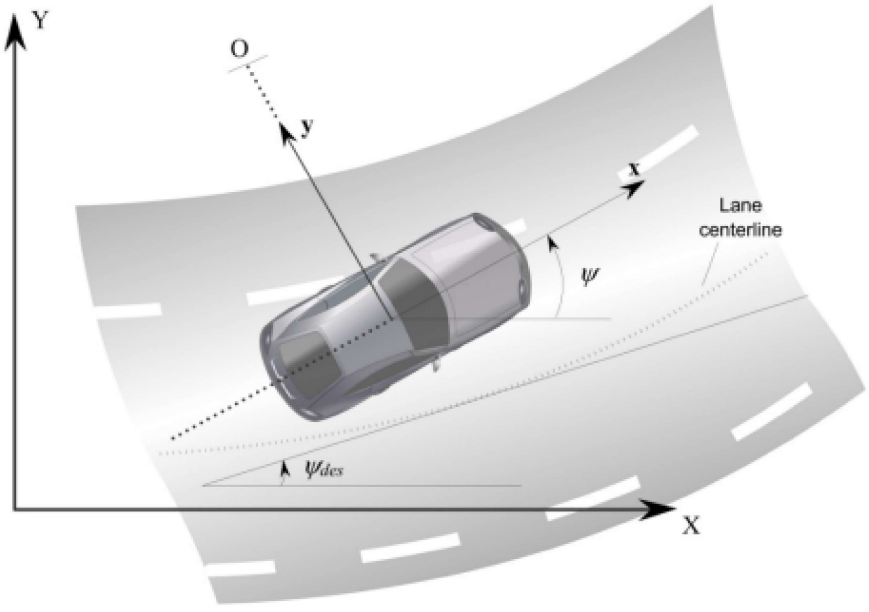


Figure 3.9: Lateral vehicle dynamics [36]

We apply newtons second law along the y-axis to obtain the relation [40],

$$ma_y = F_{yf} + F_{yr} \tag{3.62}$$

where, a_y represents the inertial acceleration of the vehicle at the c.g. while F_{yf} and $F_{yr}f$ represent the lateral tire forces in the combined wheel. The acceleration is affected by the motion in the y axis as well as the centripetal acceleration.

$$m(\ddot{y} + \dot{\psi}V_x) = F_{yf} + F_{yr} \tag{3.63}$$

By balancing the moments about the z axis we can obtain relations for yaw.

$$I_z\ddot{psi} = l_f F_{yf} + F_{yr} \tag{3.64}$$

Where the same notations represent the same parameter for the remaining terms as used in the kinematic model. Similar to how slip angle was previously defined, in lateral dynamics slip is very essential to calculating the lateral wheel forces. We represent slip angle for lateral dynamics in Figure 3.10.

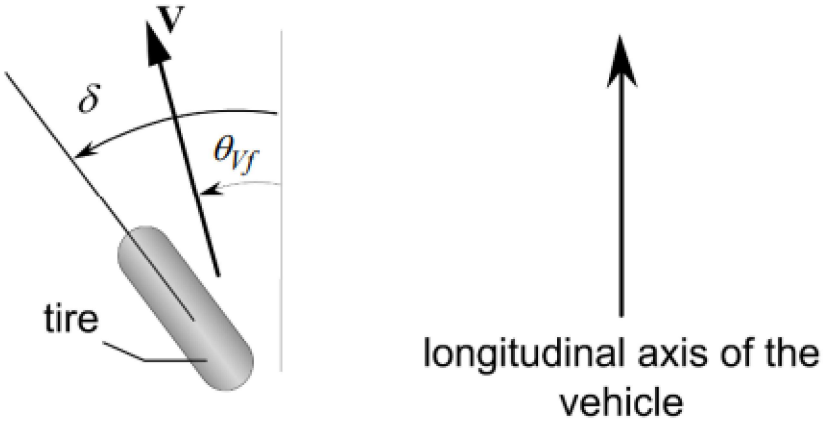


Figure 3.10: Tire slip angle

where slip angle at front wheel and rear wheel can be represented with the following

equations.

$$\alpha_f = \delta - \theta_{vf} \tag{3.65}$$

$$\alpha_r = -\theta_{Vr} \tag{3.66}$$

The lateral tire forces with the slip angles for front and rear wheel cluster can be given as,

$$F_{yf} = 2C_{\alpha f}(\delta - \theta_{Vf}) \tag{3.67}$$

$$F_{yr} = 2C_{\alpha r}(-\theta_{Vr}) \tag{3.68}$$

where the constant C_α is the cornering stiffness of each wheel. The equations are multiplied by a constant 2 since both wheels are clustered together.
Now we can resolve for θ.

$$\tan(\theta_{Vf}) = \frac{V_y + l_f\dot{\psi}}{V_x} \tag{3.69}$$

$$\tan(\theta_{Vr}) = \frac{V_y - l_r\dot{\psi}}{V_x} \tag{3.70}$$

We can take a small angle approximation to remove tan from the equations.
Finally, we can use these equations to write the state space model.

$$\frac{d}{dt}\begin{bmatrix} y \\ \dot{y} \\ \psi \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{2C_{\alpha f}+2C_{\alpha r}}{mV_x} & 0 & -V_x - \frac{2C_{\alpha f}l_f - 2C_{\alpha r}l_r}{mV_x} \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{2l_fC_{\alpha f}-2l_rC_{\alpha r}}{I_zV_x} & 0 & -\frac{2l_f^2C_{\alpha f}+2l_r^2C_{\alpha r}}{I_zV_x} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{2C_{\alpha f}}{m} \\ 0 \\ \frac{2l_fC_{\alpha f}}{I_z} \end{bmatrix} \delta \tag{3.71}$$

3.2.4 Lateral Dynamics Model w.r.t The Road

As we are trying to control the car in a urban road setting, it is advantageous to develop a mode which reflect the state variables in terms of position and orientation error with respect to the road. We take two error terms,

- e_1: the distance of the c.g. of the vehicle w.r.t the center line of lane.
- e_2: the orientation error of the vehicle w.r.t the road.

Assuming that a vehicle with constant longitudinal velocity is travelling on a road with a large radius R. Holding the small angle assumption we can define the desired rate of orientation and acceleration.

$$\psi_{des} = \frac{V_x}{R} \tag{3.72}$$

$$\frac{V_x^2}{R} = V_x \psi_{des} \tag{3.73}$$

Following the approach in [41], we now define error terms.

$$\ddot{e_1} = (\ddot{y} + V_x \dot{\psi}) - \frac{V_x^2}{R} = \ddot{y} + V_x(\dot{\psi} - \dot{\psi_{des}} \tag{3.74}$$

$$e_2 = \psi - \psi_{des} \tag{3.75}$$

$$\dot{e_1} = \dot{y} + V_x(\psi - \psi_{des}) \tag{3.76}$$

Assuming that the longitudinal velocity is constant a linear time invariant model.

$$m\ddot{e_1} = \dot{e_1}\left[-\tfrac{2}{V_x}C_{\alpha f} - \tfrac{2}{V_x}C_{\alpha r}\right] + e_2\left[2C_{\alpha f} + 2C_{\alpha r}\right] + \dot{e_2}\left[-\tfrac{2C_{\alpha f}l_f}{V_x} + \tfrac{2C_{\alpha r}l_r}{V_x}\right] \tag{3.77}$$

$$+\dot{\psi_{des}}\left[-\tfrac{2C_{\alpha f}l_f}{V_x} + \tfrac{2C_{\alpha r}l_r}{V_x}\right] + 2C_{\alpha f}\delta$$

$$I_z\ddot{e_2} = 2C_{\alpha f}l_1\delta + \dot{e_1}\left[-\tfrac{2C_{\alpha f}l_f}{V_x} + \tfrac{2C_{\alpha r}l_r}{V_x}\right] + e_2\left[2C_{\alpha f}l_f - 2C_{\alpha r}l_r\right] \tag{3.78}$$

$$\dot{e_2}\left[-\tfrac{2C_{\alpha f}l_f^2}{V_x} - \tfrac{2C_{\alpha r}l_r^2}{V_x}\right] - I_z\ddot{\psi_{des}}\left[-\tfrac{2C_{\alpha f}l_f^2}{V_x} - \tfrac{2C_{\alpha r}l_r^2}{V_x}\right]$$

The state space can now be represented in Equation 3.79.

$$\frac{d}{dt}\begin{bmatrix} e_1 \\ \dot{e_1} \\ e_2 \\ \dot{e_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{2C_{\alpha f}+2C_{\alpha r}}{mV_x} & \frac{2C_{\alpha f}+2C_{\alpha r}}{m} & -\frac{2C_{\alpha f}l_f+2C_{\alpha r}l_r}{mV_x} \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{2l_fC_{\alpha f}-2l_rC_{\alpha r}}{I_zV_x} & \frac{2l_fC_{\alpha f}-2l_rC_{\alpha r}}{I_zV_x} & -\frac{2l_f^2C_{\alpha f}+2l_r^2C_{\alpha r}}{I_zV_x} \end{bmatrix} \begin{bmatrix} e_1 \\ \dot{e_1} \\ e_2 \\ \dot{e_2} \end{bmatrix}$$

$$+ \begin{bmatrix} 0 \\ \frac{2C_{\alpha f}}{m} \\ 0 \\ \frac{2l_fC_{\alpha f}}{I_z} \end{bmatrix} \delta + \begin{bmatrix} 0 \\ -\frac{2C_{\alpha_f}l_f-2C_{\alpha r}l_r}{mV_x} - V_x \\ 0 \\ -\frac{2C_{\alpha_f}l_f^2+2C_{\alpha r}l_r^2}{I_zV_x} \end{bmatrix} \tag{3.79}$$

This creates the tracking objective of the steering control problem, however it takes the assumption that longitudinal velocity is constant.

Chapter 4

Vision Based Localization

This chapter presents the camera based algorithms used to estimate the camera pose of the UAV in order to help update its state estimates and construct a map of the world. The algorithms are also needed in order to localize the UGV within the camera frame of the UAV and hence, localize it within the aerial view map.

4.1 Pose Estimation

The pose estimation problem for the UAV is tackled through a filter based visual inertial solution. For visual estimates. The pose estimation strategy follows such that the UAV has a birds eye view of the ground and captures new frames consecutively an example of this frame capture strategy is presented in Figure 4.1. The 3D motion information is matched by extracting and tracking feature points in the consecutive frames.

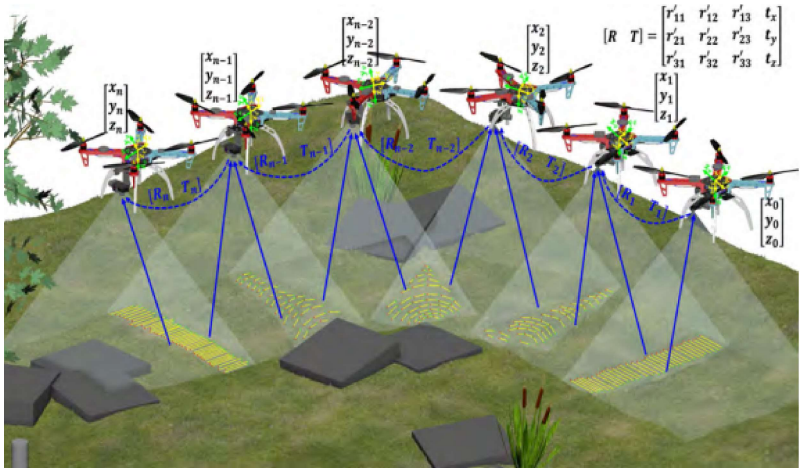


Figure 4.1: Pose Estimation Strategy, where R and T represent rotation and translation matrices [42]

A high number of unique feature points being detected in consecutive frames is desirable for accurate motion estimation.

4.1.1 Feature Detection and Matching

Feature detection is a low level image processing techniques which identifies unique points of interest that ideally are replicate-able from different view points. Corner points are one such feature that can be detected through algorithms such as the Harris corner detection [43]. While feature description refers to the method of describing these points of interest with reference to the pixels around them. Today many algorithms such as SIFT, SURF, FREAK, BRIEF, and ORB exist that can both detect features and describe them [44], [45]. These detectors are widely applied to visual navigation purposes.

Popular choices among these algorithms are the SIFT, SURF and ORB features:

1. SIFT: Scale Invariant Feature Transform features were one of the first modern feature detectors presented to extract points of interest from images. SIFT algorithm follows 4 main steps: 1. Use difference of Gaussian to identify key points from a pyramid of scale, localise important key points, orientation assignment and key point description.

2. SURF: Speed-Up Robust features was an algorithm that was inspired by the SIFT algorithm. The SURF algorithm sped up the process of feature detection by making use of integral images [46]. The SURF method is based on two major steps: Using Laplacian of Gaussian on images followed by the Hessian matrix for identifying key points and key point description.

3. ORB: Oriented Fast and Rotated Brief is an algorithm that took advantage of two other algorithms, FAST and BRIEF. ORB makes use of FAST for feature detection and BRIEF for feature descriptions. ORB features have proven to provide the same accuracy as SIFT while being two orders of magnitude faster. For these reasons we chose ORB as the primary algorithm for feature detection and explain it in greater detail below.

To understand ORB we first look at FAST feature detection. FAST works by computing the brightness of a pixel p, such as that given in Figure 4.2, with the surrounding 16 pixels. The pixels in the neighbourhood of p are then sorted into three categories: Darker than p, same as p and lighter than p. If more than 8 pixels are either darker or brighter than p it is selected as a key point [47].

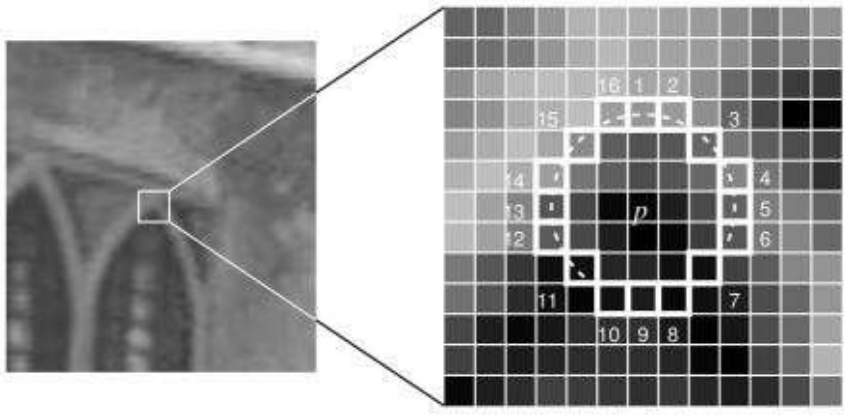


Figure 4.2: FAST algorithm detecting a key point

However, unlike SIFT or SURF, FAST features do not have an orientation or multi scaling component. Which means that for different orientations or scaling of the image FAST may not be able to find the same keypoints. To ameliorate this ORB

algorithm implements a multiscaled image pyramid (see Figure 4.3). This pyramid consists an array of different scales of the same image. Each level contains a down sampled resolution. After creating an array of down-sampled images, ORB finds keypoints based on FAST at every resolution level, making orb partial scale invariant.

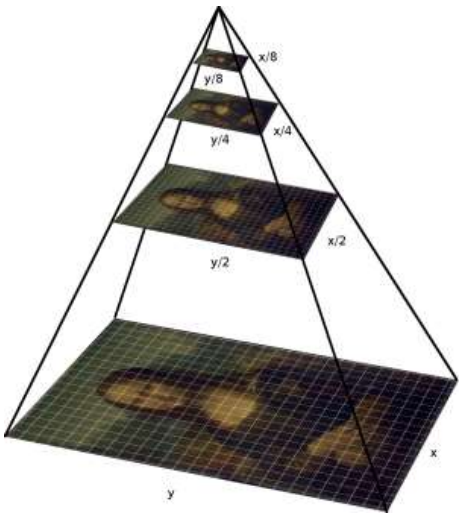


Figure 4.3: Pyramid of Scales

After the detection of keypoints, an orientation to each of them is defined based on the levels of intensity around the pixel of interest. This action is carried out through intensity centroids achieved by first calculating moments of a patch given in Equation 4.1.

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y) \tag{4.1}$$

The center of mass of the patch can easily be identified with Equation 4.2.

$$C = (\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}) \tag{4.2}$$

Now a vector from the center to the centroid can be calculated.

$$\theta = \arctan 2(m_{01}, m10) \tag{4.3}$$

With the orientation calculated, the patch can be rotated to a canonical rotation and description can be carried out to achieve rotation invariance. A visualization of this is given in Figure 4.4.

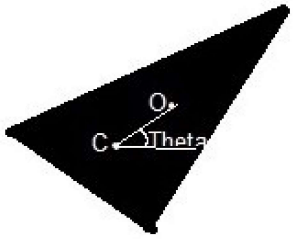


Figure 4.4: A canonicaly rotated patch

Once the key points are sought out by the FAST detector, BRIEF is used to convert the key points into binary feature vectors. The binary feature description is a feature vector that only contains values of 0 and. This part starts by smoothing the image using a Gaussian Kernel to prevent the descriptor from failing to high frequency noise. However, BRIEF is not invariant to rotation. Hence, ORB makes an adjustment by providing a patch orientation θ and the corresponding rotation matrix, this operation basically steers the BRIEF operator. It finally discretizes the angle to increments of 12 degrees and constructs a lookup table of BRIEF patterns. As long as the steered BRIEF remains consistent across views, BRIEF will not fail to rotation.

4.1.2 Processing and Outlier Detection

Having described how our detector works, for any image we first calculate the relevant features and extract its descriptors using orb. These features are then matched against the following frame to capture the motion.

The features are matched using fast library for approximate nearest neighbours (FLANN) which is comparatively faster than the brute force algorithm. The match-

ing process is obtained through a library of nearest point matches on the FLANN matcher. For improving accuracy it is important to remove points whose distance from closest to second closest is greater than an assigned distance threshold. The filtered match points can be presented as:

$$p(n) = \begin{cases} \text{(x,y,s)}, & \text{if } distance - ratio < 0.3 \\ 0, & \text{otherwise} \end{cases}$$

where s represent the size of the keypoint and distance ratio is the minimum ratio of the matched points.

While these start to allow us to make better approximations, extracted points are still very susceptible to outliers that can be generated due to poor images, motion blur, dynamic objects, or optical effects. The existence of outliers can greatly affect the result and cause significant error in the motion.If the UAV loses track of its states than UGV will be localized incorrectly which can cause great errors. Hence, the RANSAC algorithm can be employed in order to generate an ideal solution [48]. Using available observations the RANSAC can predict and remove outliers greatly improving the number of unique key points in the map. Figure 4.5 shows a RANSAC algorithm differentiating between inliers and outliers

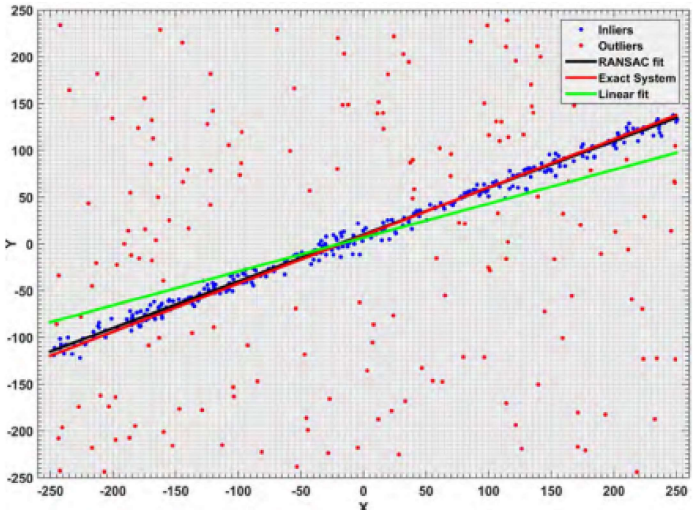


Figure 4.5: RANSAC Algorithm Predicting Outliers

As an additional step before we pass our image to the feature detector we carry out some necessary pre-processing particularly for illumination correction. As a camera generally works with visual properties such as contrast, in order to deal with optical effects from the environment. To achieve this we can increase the dynamic range of the image intensities by using the histogram of equalization, given in Equation 4.4.

$$I_{hist} = \frac{L(cdf_i - cdf_{min} - (N \times M)}{N \times M} \tag{4.4}$$

Where L represents the number of gray levels, N,M are the image dimensions, while cdf_{min} is the minimum non zero value of the cumulative distribution.
This action is followed by applying gamma correction.

$$I_{cor} = I_{hist}^{\frac{1}{\gamma}} \tag{4.5}$$

Where gamma tends to be less than one for darker images and white for brighter. The results of such a correction is depicted in Figure 4.6.

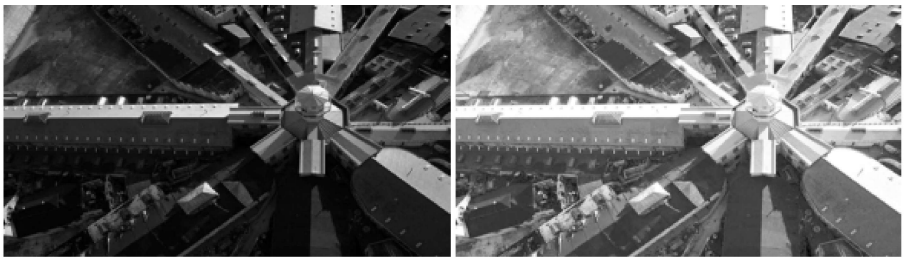


Figure 4.6: Example of Illumination Correction

4.1.3 Homography Formulation

Homography describes a planar relationship that is used to define points from one plane to another. In homography a 3 × 3 matrix transforms 3 dimensional vectors that represent 2D points on the plane as shown in Figure 4.7. The vectors are described as the homoegeneous coordinates. The homography matrix can be decomposed to retrieve the orientation and position of the camera.

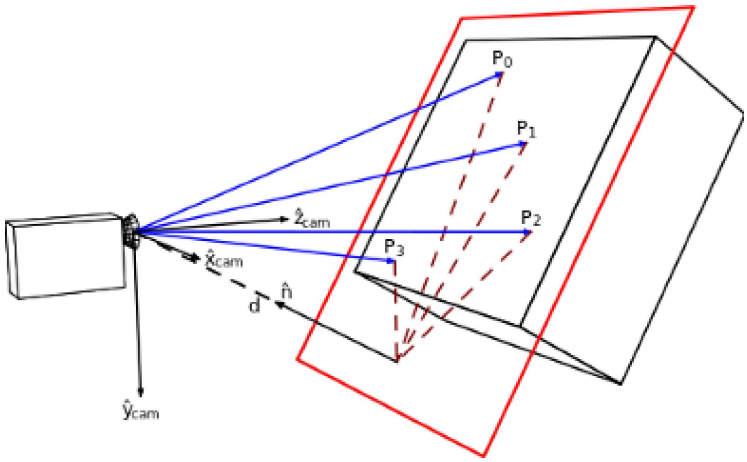


Figure 4.7: Shows how 4 points on the plane relate to the image plane

Again, homogeneous coordinates are projective points which serve a key role in computer vision. Since, the image plane is 2D when, the dimensionality of depth is lost. Which means that any number of 3D points can be projected on a single pixel location on the image plane.

We first describe the pin hole model to better explain how homography is a special case for the model.

The pin hole camera can be defined through its projection matrix. Which is essen-

tially a single matrix formed from two other matrices that relate to the properties of the camera. The projection matrix is in the form of Equation 4.6.

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \tag{4.6}$$

The extrinsic matrix stores information relevant to position of the camera in the world frame. This information is stored as a rotation and translation matrix, storing the cameras 3D orientation and translation respectively.

$$M_{ex} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \tag{4.7}$$

Where the values r represent rotation and t represent translation. Since the extrinsic matrix maps homogeneous coordinates from the global frame, all transformed vectors will represent the same position with respect to the focal point.

Another matrix that is used to define a cameras inherent properties is the intrinsic matrix. The intrinsic matrix stores properties such as the focal length and principal points of the camera. The intrinsic matrix transforms the 3D coordinates relative to the focal point to the image. The intrinsic matrix is given as:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.8}$$

When both these matrices are combined we obtain the pin hole camera model.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = KM_{ex} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{4.9}$$

The homography matrix is simply just a special case of the pin hole model where all the projected coordinates on the camera are lying on a plane such that the z coordinate is 0. Such that our pinhole model follows conversion from Equations 4.10 to 4.12.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = KM_{ex} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} \tag{4.10}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ r_{31} & r_{32} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{4.11}$$

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{4.12}$$

Now the homography matrix can be given as H and it give a matrix that defines the transformation of points from one plane to another. The homography matrix makes use of the 4 point algorithm [49] to find correspondences from one plane to the next.

Now we had calculated the ideal feature points through RANSAC and we can find the error of two matched points using symmetric transfer error from homography H

as,

$$d^2_{transfer} = d(p_i, H^{-1}p_{i-1})^2 + d(p_{i-1}, Hp_i)^2 \tag{4.13}$$

where p and $p_i - 1$ are the matched points and d refers to the distance between them.

Taking points p_{i-1} from a previous frame f_{i-1} and taking correspondences with current point p and frame f we can extract the frame to frame motion from image plane homography.

$$sp_i = H^i_{i-1}p_{i-1}$$

$$s\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_{i-1} \\ y_{i-1} \\ 1 \end{bmatrix} \tag{4.14}$$

We can finally carry out decomposition using SVD on the homography matrix to find the rotation matrix represented by the first two column and the translations represented by the last column of the decomposed matrix.

4.2 Object Detection

Since the camera on the UAV is set to look downwards towards the ground to detect both the features in the environment as well as the vehicle, it is necessary to remove the features detected by the camera on top of the UGV. Since the UGV is a dynamic body in the camera image, its continuous presence can result in the algorithm incorrectly updating the camera pose. Hence, it is suggested that a fast object detection algorithm is run which can quickly detect where the UGV is in the frame and pass this information to the feature detector in order to make the necessary corrections. We propose using a famous and light weight neural network for this task known as Yolo [50].

Yolo or You Only Look Once is a re-imagined method of detecting obstacles us-

ing machine learning. Conventional object detection techniques in machine learning involve detecting obstacles by comparing it against classifiers of the given object and test it at various parts of an image by re-scaling it. This process can involve revisiting the same pixel values multiple times for each segment of the algorithm and often to pre and post process them for classification, this redundancy of the operation can result in slower outputs. Instead, Yolo seeks to carry out the entire process using a single convolution network which detects all bounding boxes and given class probabilities for those boxes by processing the image only once. An initially proposed model which was later optimized for speed using anchor boxes is described in Figure 4.8.

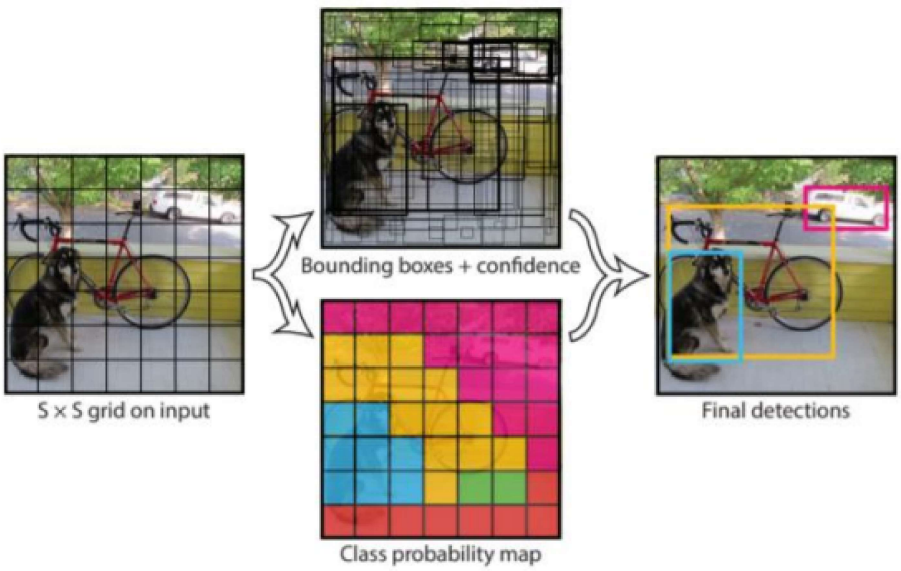


Figure 4.8: Yolo System Model

The performance power of Yolo comes from its ability to process images globally instead of using a sliding window or region-based techniques. This allows Yolo to implicitly factor in contextual information about classes and their appearance without them having to be fed separately. The class-specific scores for each box is calculated via the following formulae:

$$P(Class_i|Object) * P(Object) * IOU_{pred}^{truth} = P(Class_i) * IOU_{pred}^{truth}$$

These scores are representative of both the probability of a class appearing in the respective box as well as how accurately the predicted box is fitting over the object [51].

While the model provided in aforementioned figıre can provide decent live processing results, it is considerably inefficient since the bounding boxes over the entire image have to be processed. Hence, an improvement on this primary model was made in [52], which makes use of dimension clusters as anchor boxes. Anchor boxes are class specific boxes that are generally defined and tuned by the user according to what the object detection algorithm is purposed to do. Anchor boxes are fundamentally different from bounding boxes such that, anchor boxes are fed to the network before training while bounding boxes are regions where the system believes an object might exist.

4.2.1 Anchor and Bounding Boxes

Anchor boxes serve as a form of maximum and minimum standard against which all detected bounding boxes are compared against. One standard of comparing a bounding box against an anchor box is the Intersection of Unions standard. The IOU standard fits different aspect ratios of bounding boxes to the anchor box. The better the fit the higher the score. A threshold is used to either identify a bounding box identifying an object or not. If an IOU score is greater than 50 percent an object 8 has been detected if it is lower than 50 percent the system identifies it as a place without an object.

In case of Yolo, the anchor boxes are not defined by the user but are instead calculated using k-means. The data set we used already contains the relative objects bounded by rectangles according to their center coordinates, width and height. These bounding boxes are then passed to a k-mean clustering algorithm to automatically find good priors [52]. When predicting, the coordinates of the bounding box are calculated as offsets from the anchors. These predicted bounding boxes are

adjusted to the anchor according to the IOU and the anchors that best fit these bounding boxes are used to classify the object. The parameters of these bounding boxes and how they correspond to anchor boxes are described in Figure 4.9.

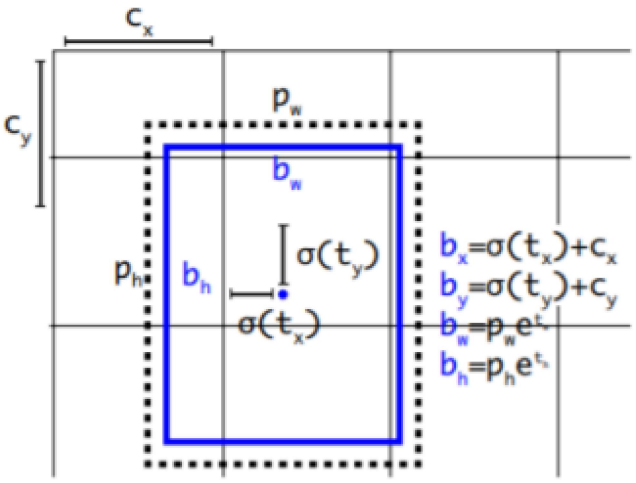


Figure 4.9: Bounding Box Parameter Determination

4.2.2 Loss Function

An integral part of the Yolo algorithm is its loss function. The loss function makes use of sum-squared error between predictions and ground truths to calculate loss. The complete loss function serves to calculate the classification loss, the localization loss and the confidence loss.

The localization loss in Yolo is defined as follows:

$$\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}[(x_i-\hat{x_i})^2+(y_i-\hat{y}_i)^2]+\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}[(\sqrt{w_i}-\sqrt{\hat{w_i}})^2+(\sqrt{h_i}-\sqrt{\hat{h_i}})^2]$$

The localization loss is responsible for measuring the difference between the ground truth and predicted location and size of boundary box. Next the confidence loss is defined as:

$$\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}(C_i-\hat{C}_i)^2+\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{noobj}(C_i-\hat{C}_i)^2$$

Table 4.1: Localization loss Parameters.

Parameter	**Description**
1_{ij}^{obj}	Value given to cell responsible for object in its space
λ_{coord}	Weight for loss in given boundary box coordinate
x_i	Horizontal pixel of centroid for anchor box
y_i	Vertical pixel of centroid for anchor box
w_i	Width of box
S	Number of cells
B	Number of anchor boxes

Table 4.2: Confidence loss Parameters.

Parameter	**Description**
1_{ij}^{noobj}	Value given to cell that detects background
λ_{noobj}	Weight loss for background
i	represents box confidence score of box at given (i , j) coordinate.

The confidence loss measures the objectness of the box i.e. it detects whether the boundary boxes are correctly identifying objects or empty spaces according to criterion. Since there will always be more empty spaces in an image where objects do not exist, a class imbalance might occur. λ_{noobj} is hence tuned to ameliorate this problem by fudging the loss down (by default its value is 0.5). Finally, we have the classification loss, which represents predicting the wrong label in a given box. It is given by:

$$\sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \sum_{classes} (p_i(c) - \hat{p_i}(c))^2$$

The loss output can be summarized as a vector of the form [50]:

$$S \times S \times B \times (4 + 1 + C)$$

The loss function serves to minimize detection of boxes which do not have any objects within them as well as to improve the accuracy of the system over each iteration. Since, the utilization of this loss function makes the algorithm unsupervised a correctly labelled data set, and the richness of the data set can affect algorithm

outcomes.

4.3 State Estimation

Inertial measurement units are self contained sensors that use internal gyroscopes, accelerometers and magnetometer to detect linear acceleration, rotation rate and heading. Inertial measurements provide the benefit of robust outputs at a high frequency but suffer from internal biases and noise. Since position estimates from an IMU are integrated values, any errors in the readings can accumulate over time which can cause the IMU to drift. The distance between the actual values and estimate values will keep increasing unless the drift is removed, to accomplish this IMUs are often used with an external sensor whose values are combined with the IMU readings through a fusion algorithm. Kalman filter is one such algorithm which is widely used for state estimation [53].

In fact, a common implementation of IMUs known as the strap down inertial system, also makes use of a Kalman filter. In the strap down implementation, the gyroscope is used to calculate angular rates which are integrated for angular position. This data is then fused with gravity vector outputs from the accelerometer to find attitude estimates. The measurements from the accelerometer are transformed into the inertial reference frame using the attitude estimates. These accelerations can now be integrated to get the linear velocity and the linear position.

4.3.1 Kalman Filter

The purpose of the Kalman filter is to continuously update state estimates of interest from noisy measurements from sensors, and for this particular case, from the IMU and from visual odometry. The approaches to filter based sensor fusion can be divided into two categories:

- Loosely coupled

- Tightly coupled

The type of coupling decides how the data will be fed into the filter. In a loosely coupled approach the data from both the IMU and camera will have been pre-processed before being sent for Kalman filtering, while in the latter approach the data is passed to the filter without any pre-processing. Figure 4.10 shows the flow of these approaches.

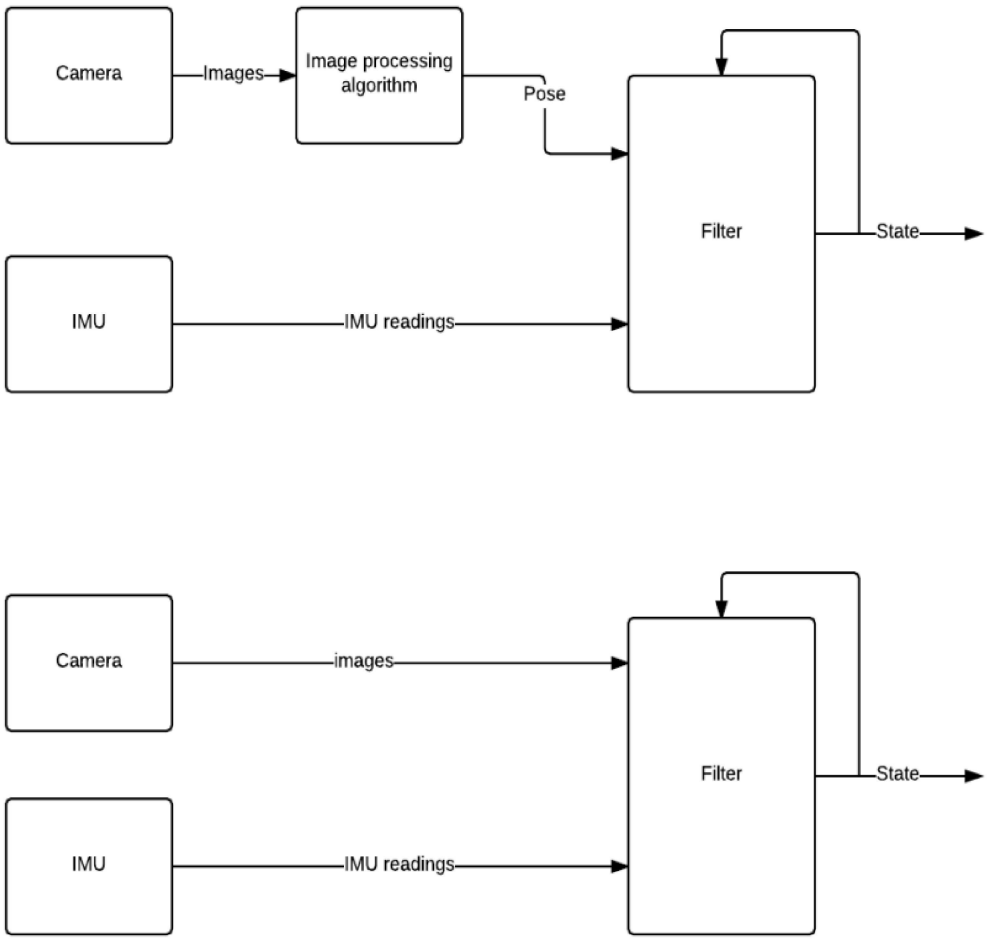


Figure 4.10: Loosely (above) and tightly (below) coupled architectures for sensor fusion [54]

Since the comparison between pre-processed measurements is computationally less expensive, the loosely coupled integration approach is taken for our estimation task.

The filter of choice for our system is the Extended Kalman Filter (EKF) which was designed to extend the application of the original Kalman filter for non-linear

systems. We first define the state space form that needs to be modeled to employ the EKF.

$$x_{k+1} = f(x_k, u_k, v_k, T) \tag{4.15}$$

$$y_k = h(x_k, u_k, e_k) \tag{4.16}$$

Where, x_k is the state at time or observation k, u_k is the input, v_k is the process noise, T is sampling interval, y_k is the measurement at time t and e_k is the measurement noise. Equation 4.15 describes the motion model of the system and Equation 4.16 describes the measurement model.

The noise terms v and e are modeled as zero mean multivariate Gaussian noises. Since, the state space in our system is non-linear the states and measurements may not necessarily be jointly Gaussian. Regardless, the probability distribution function of the states is still approximated with Gaussian disturbance. The motion model is approximated using a first order Taylor expansion which is then applied to the standard Kalman filter routine described below.

In the Kalman loop the first step following initialization is the prediction time update. In this step the states and inputs of the system are propagated through the motion model while the covariance of the states are updated by adding uncertainty due to process noise. This step is described in the equations:

$$\hat{x}_{k+1|k} = f(\hat{k}_{k|k}, u_k, \hat{v}_k, T) \tag{4.17}$$

$$\hat{P}_{k+1|k} = F_k P_{k|k} F_k^T + L_k Q_k L_k^T \tag{4.18}$$

where,

$$F_k = \frac{\delta f}{\delta x} \mid_{\hat{x}_{k|k}, u_k, \hat{v}_k, T}$$

$$L_k = \frac{\delta f}{\delta v} \mid_{\hat{x}_{k|k}, u_k, \hat{v}_k, T}$$

$\hat{x}_{k+1|k}$ is the predicted state, $\hat{x}_{k|k}$ is the filtered state, $\hat{v}_k$ is the expected value of noise and Q_k is its covariance. The covariance describes the errors in the model and is measure of how much the motion model can be trusted.

Next, the measured update is compared to the observed measurement using the predicted measurement. This allows the optimal gain to be calculated and the proportional error to be updated. This is described in a set of equations as:

$$K_k = P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + M_k R_k M_k^T)^{-1} \tag{4.19}$$

$$\hat{x}_{k|k} = \hat{k}_{k-1} + K_k(y_k - h(\hat{x}_{k|k-1}, u_k, \hat{e}_k)) \tag{4.20}$$

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} \tag{4.21}$$

where,

$$H_k = \frac{\delta h}{\delta x} \mid_{\hat{x}_{k|k}, u_k, \hat{e}_k}$$

$$M_k = \frac{\delta h}{\delta e} \mid_{\hat{x}_{k|k}, u_k, \hat{e}_k}$$

$\hat{e}_t$ is the expected value of measurement noise, while R_t is the covariance of e_t.
For the EKF to properly work it is important for initial guesses $x_{0|0}$ and $P_{0|0}$ to be accurate. From our simulation, these values can be directly passed to the filter from the environment. The Kalman filter process is summarised in Figure 4.11.

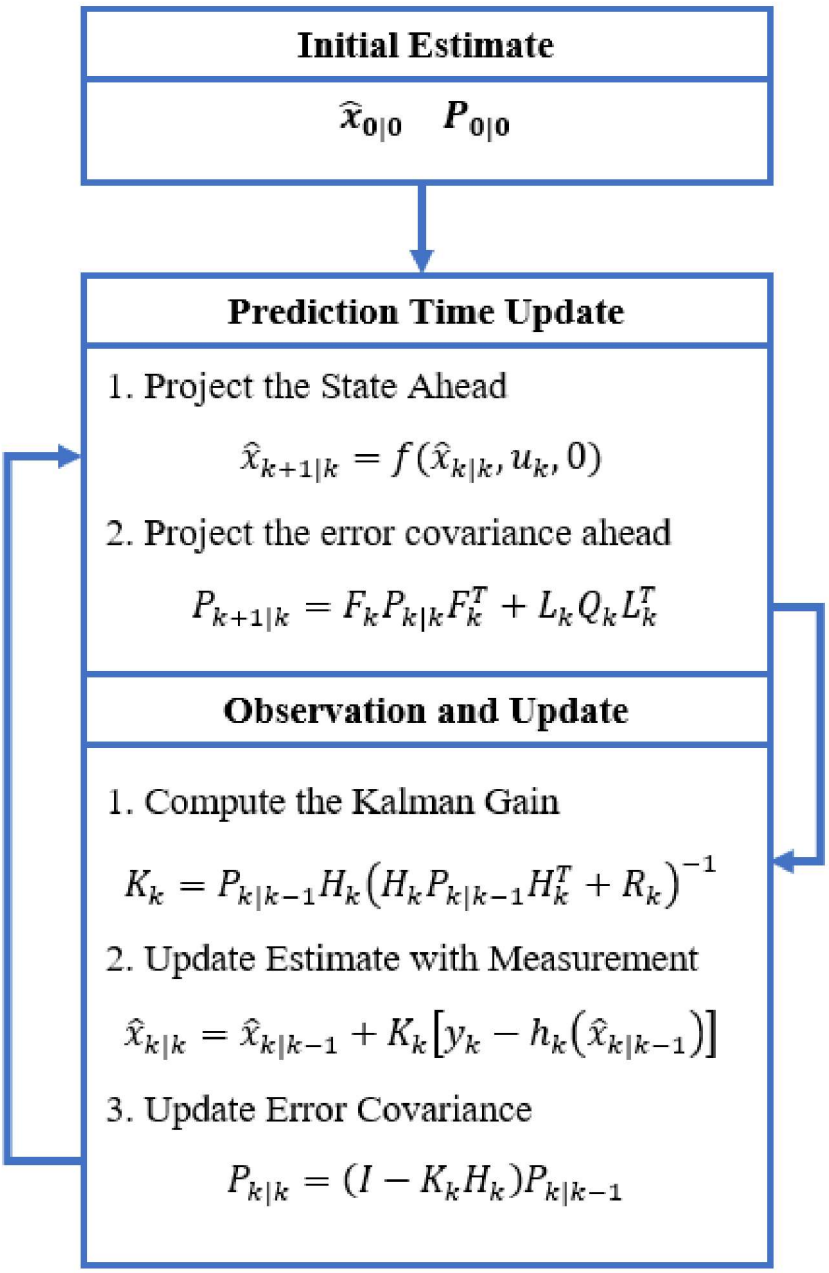


Figure 4.11: Extended Kalman Filter Algorithm

Chapter 5

Simulations and Results

In this chapter we describe how the simulations are set up and how the described models behave and the estimations of the UGV are carried out. We describe the following tasks here:

- Implementation and testing of Stanley Controller to observe how the modeled system behaves when given tracked points.
- How the computer vision algorithms work to observe UGVs in the image frame.
- Estimation of UGV states using a Kalman Filter on the UGV which receives updates from the UAV for sensor fusion.

5.1 UAV Simulation

Based on the equations discussed in Chapter 3 a SIMULINK model of the quadrotor is prepared in MATLAB. As Carla does not inherently support UAVs, a flying camera which exhibits the dynamical properties of a quadrotor is simulated in the environment. The camera moves such that before every time step within the Carla world, the MATLAB engine computes the transformations that the camera will experience and passes them through the Python API. Generally, Carla creates its own dynamic atmospheric condition however, these can not be passed efficiently or easily

to MATLAB. Hence, for the quadrotor simulations the COESA atmosphere model is used which directly translates to the ICAO atmosphere model under 32,000 feet. This is a reasonably good model for low altitude simulations. In Figure 5.1 the SIMULINK model is shown, while Table 5.1 shows the parameters used.

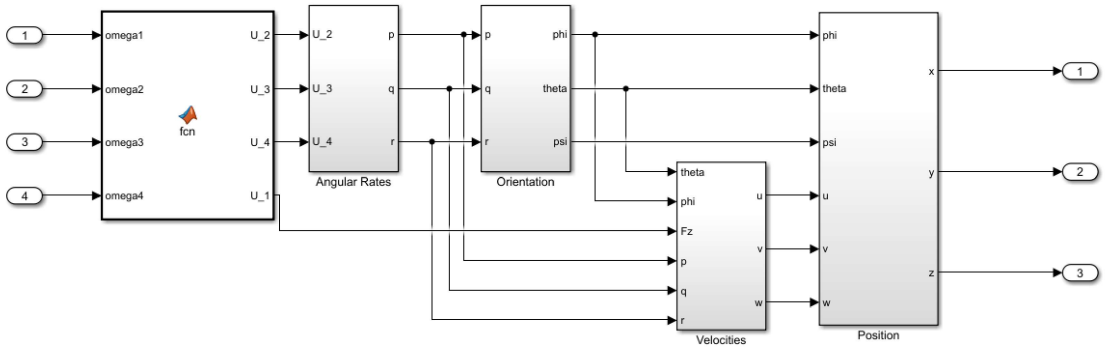


Figure 5.1: Simulink Model of UAV.

Table 5.1: Simulation Parameters

Parameter	**Description**	**Value**
m	Mass	1 kg
J_r	Inertia of Motors	$6x10^{-7} kg.m^2$
I_{xx}	Roll Moment of Inertia	0.005 kg.m^2
I_{yy}	Pitch Moment of Inertia	0.005 kg.m^2
I_{zz}	Yaw Moment of Inertia	0.0095 kg.m^2
A_D	Aerodynamic Area	0.0025 m^2
C_D	Drag Coefficient	0.5
l	Arm length	0.15 m
d	Drag Factor	$7.7x10^{-7} N.m.s^2$
b	Thrust Factor	$3.13x10^{-}5$

In order for the quadrotor to make meaningful movements within the simulation environment it is necessary for it to be able to follow the desired path. Hence, a cascading PD control is designed for position and attitude responses, while a PID is defined for controlling the height of the UAV. The architecture of the control is given in Figure 5.2.

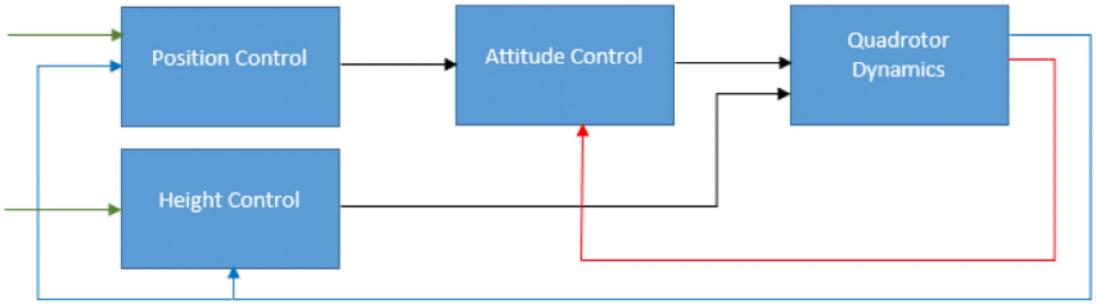


Figure 5.2: UAV Control Architecture

Since it is important for UAV to maintain a constant height for most of the simulation, the UAV is passed a command to increase its height, with its PID response given in Figure 5.3.

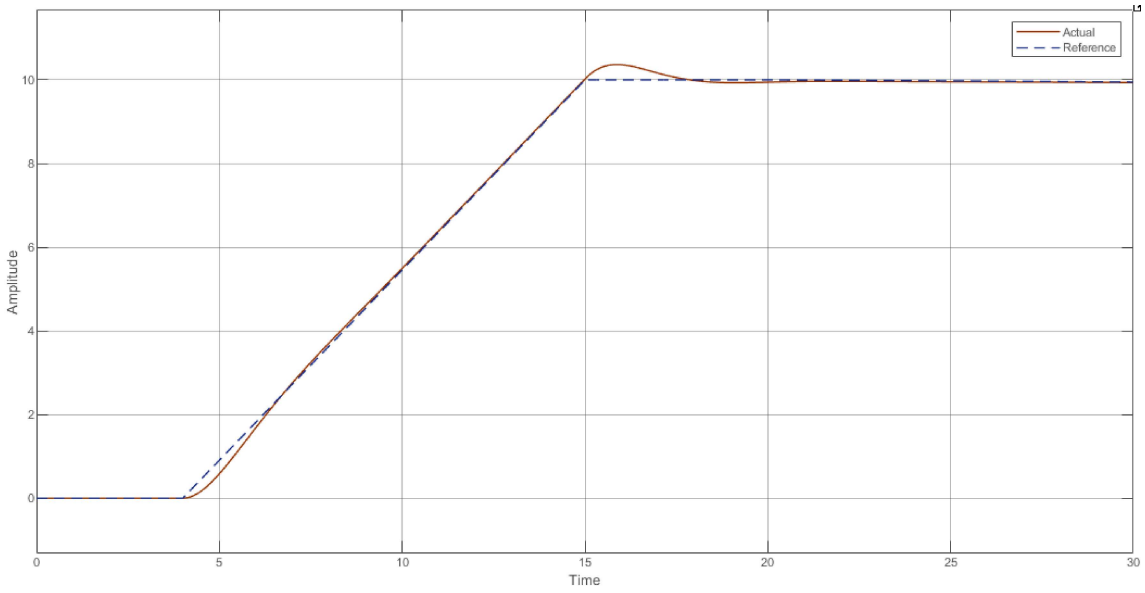


Figure 5.3: Height Tracking

The height control shows reasonable response. Next the UAV is tested against different trajectories, based on difficulty and control demands. The first trajectory is a rectangular trajectory, Figures 5.4 to 5.8 show UAV responses.

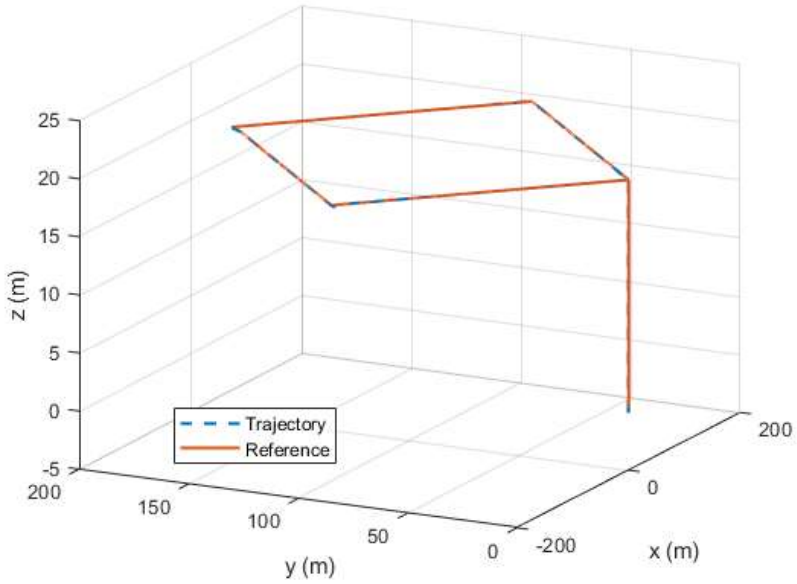


Figure 5.4: Rectangular Trajectory

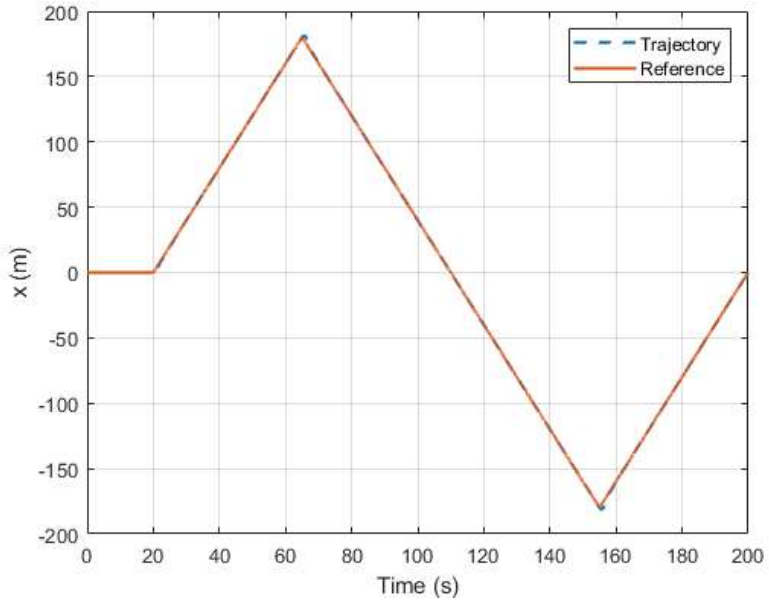


Figure 5.5: x Position Response to Rectangular Trajectory

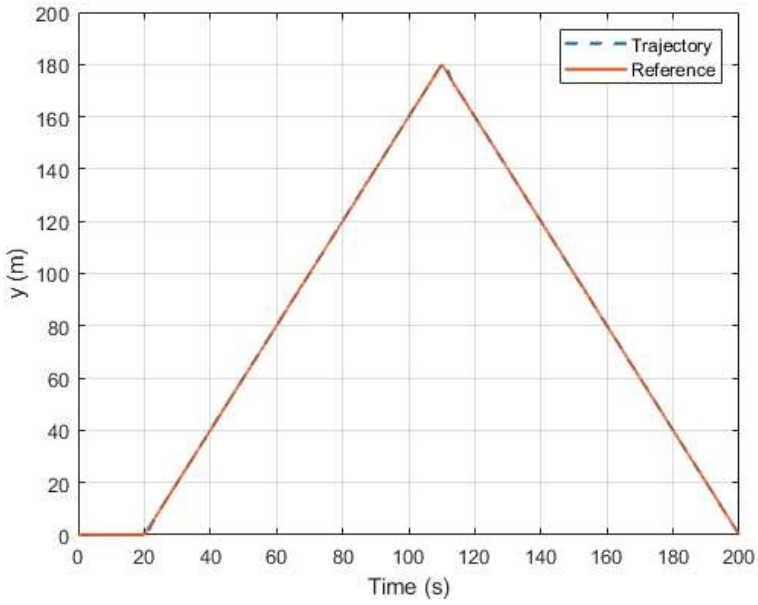
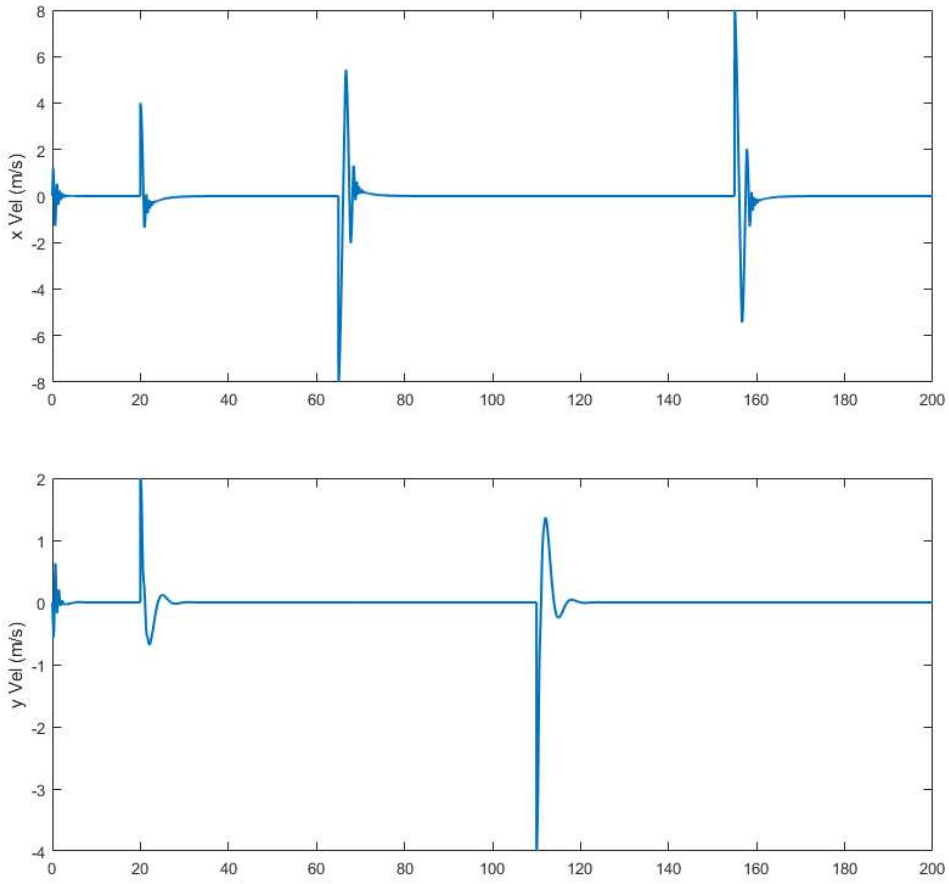


Figure 5.6: y Position Response to Rectangular Trajectory



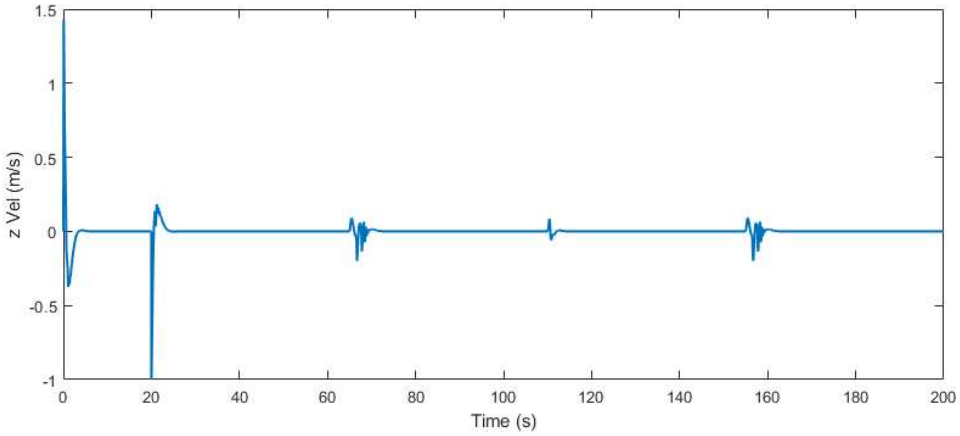


Figure 5.7: Velocity Errors in Rectangular Trajectory

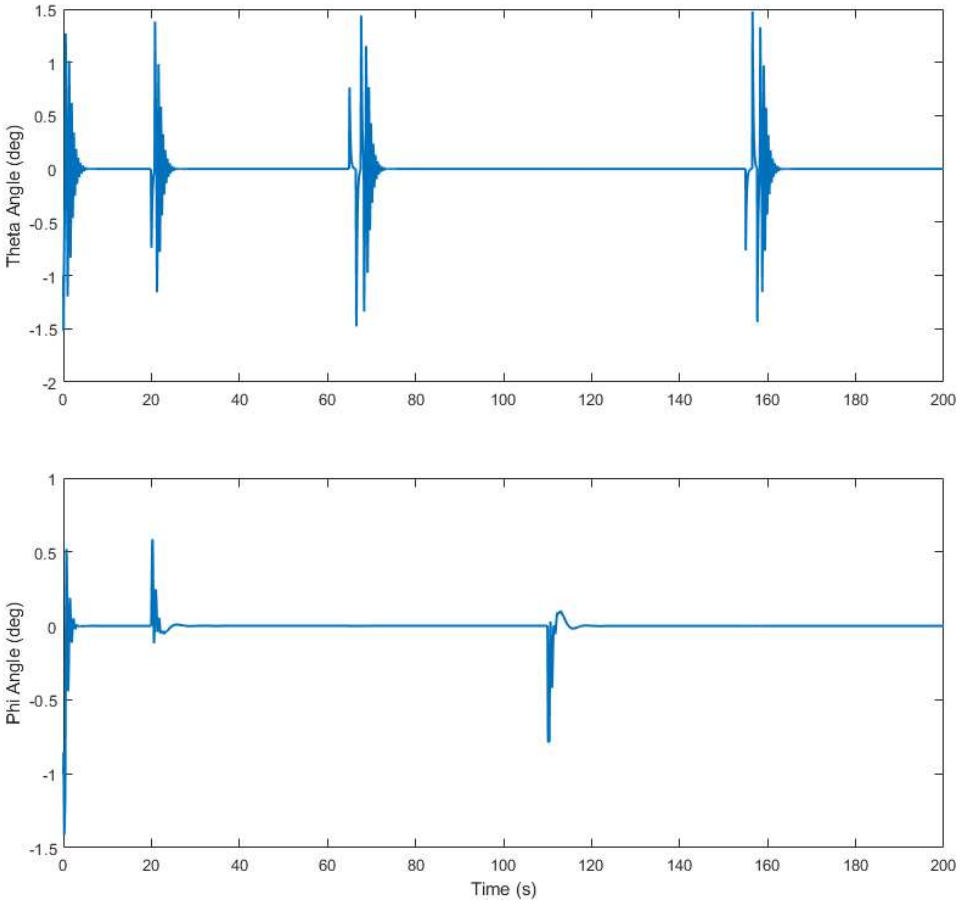


Figure 5.8: Angular Errors in Rectangular Trajectory

From the responses we can identify that the UAV has a jerk reaction on initialization, which can be attributed to the aerodynamic forces acting on the quadrotor that the controller immediately tries to overcome. For the remaining responses the UAV

Table 5.2: Error Parameters Rectangular Trajectory

Parameter	**RMS Error**	**Parameter**	**RMS Error**
x (m)	0.59	$\dot{y}$ (m/s)	0.32
y (m)	0.25	$\dot{z}$ (m/s)	0.10
z (m)	0.05	$\theta(deg)$	0.20
$\dot{x}$ (m/s)	0.92	$\phi(deg)$	0.11

shows acceptable results even though the RMSE for x position and velocity responses are high this is due to the sharp corners that the UAV has to take in this trajectory. The rectangular trajectory is the closest depiction of the waypoints the UAV will be expected to pass through when detecting UGV in the simulation map.

Next we take a look at how the UAV performs against an ascending spiral trajectory, the results are depicted through Figures 5.9 to 5.13.

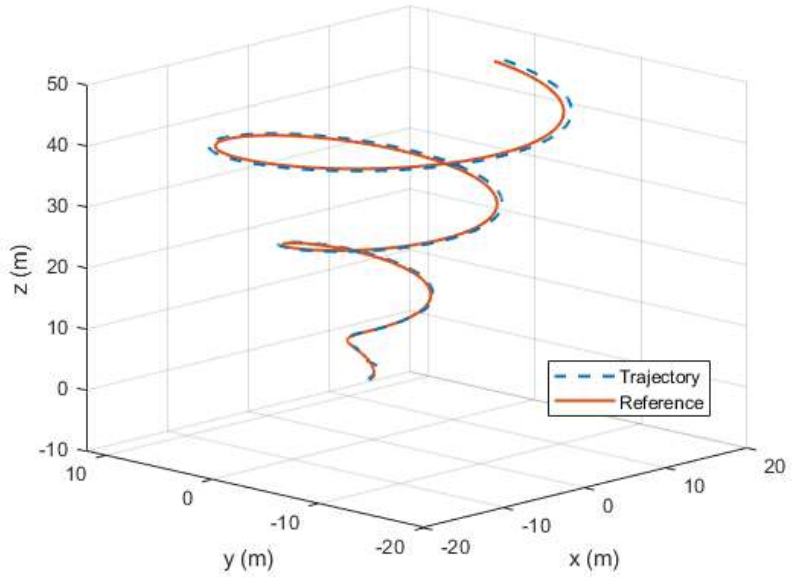


Figure 5.9: Ascending Spiral Trajectory

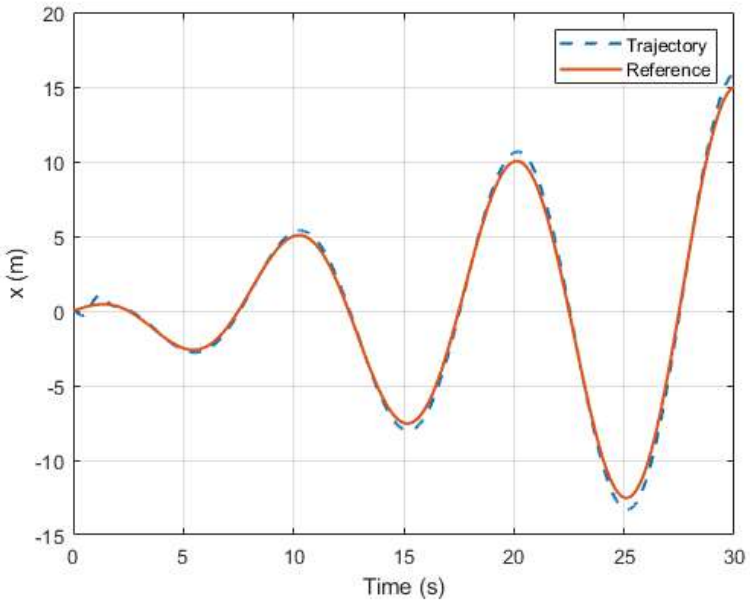


Figure 5.10: x Position Response to Spiral Trajectory

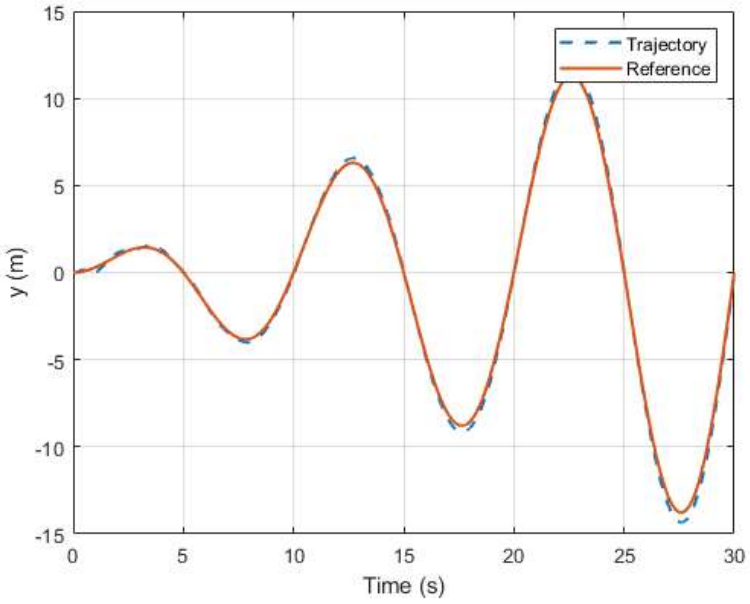


Figure 5.11: y Position Response to Spiral Trajectory

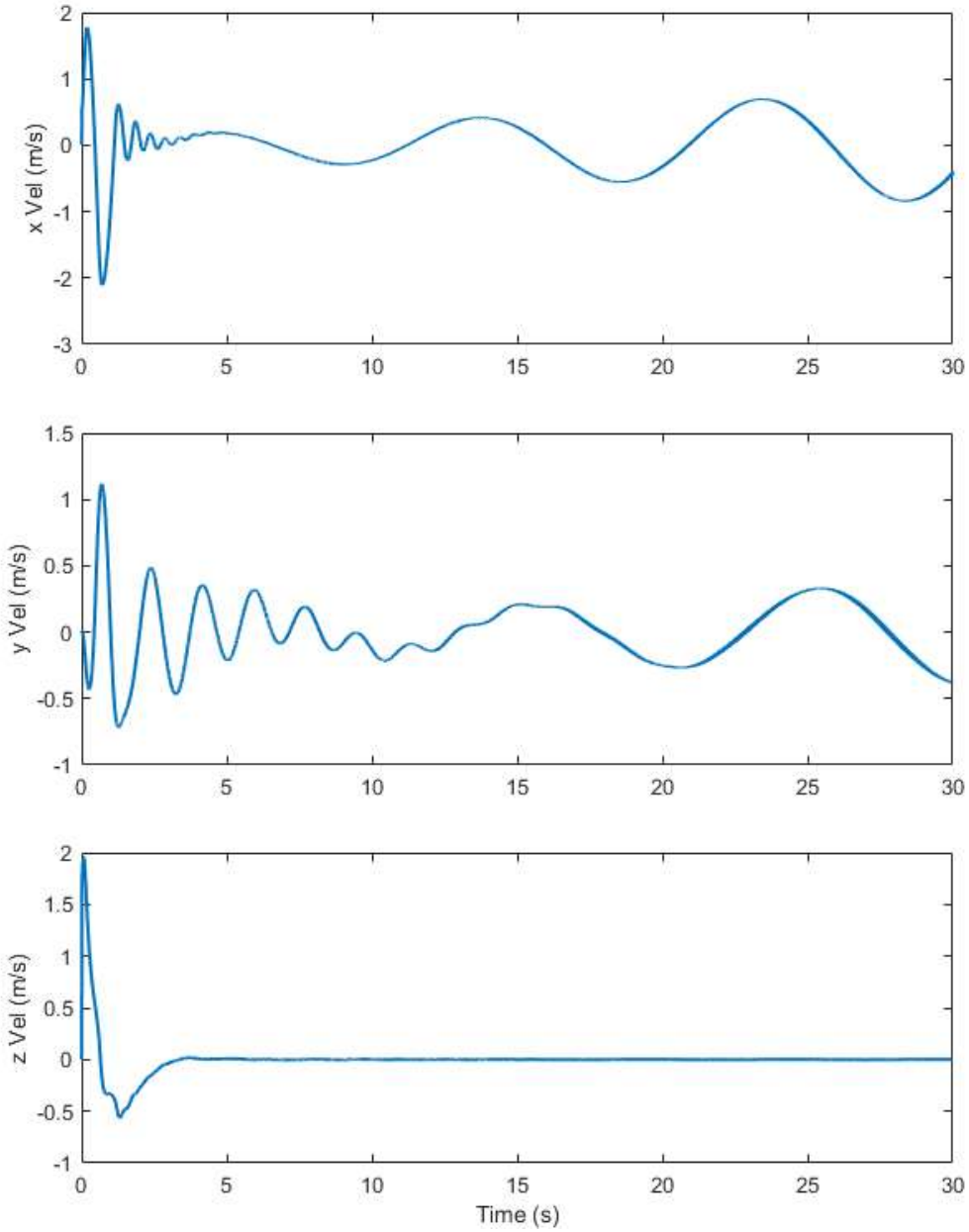
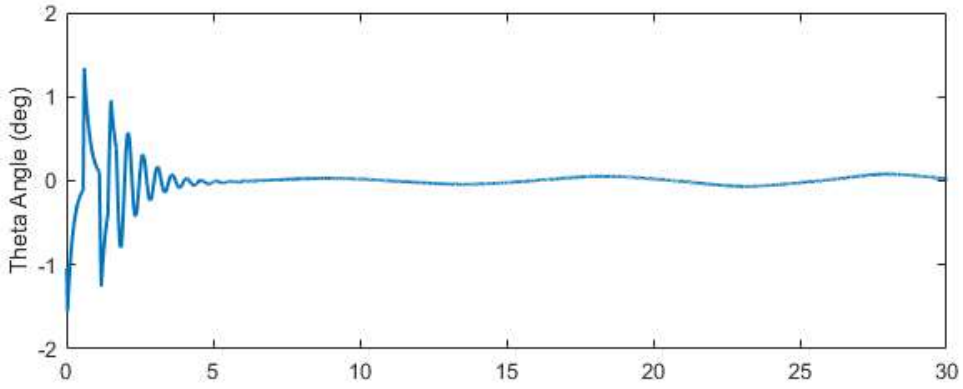


Figure 5.12: Velocity Errors in Spiral Trajectory



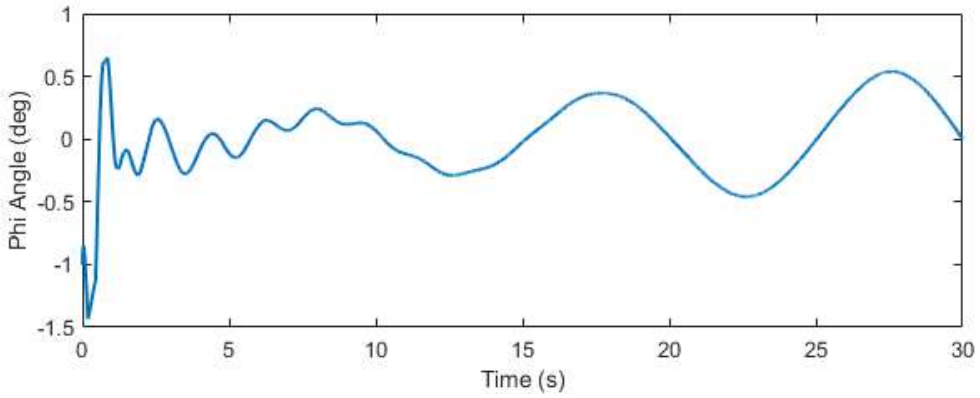


Figure 5.13: Angular Errors in Spiral Trajectory

Table 5.3: Error Parameters Spiral Trajectory

Parameter	**RMS Error**	**Parameter**	**RMS Error**
x (m)	0.60	$\dot{y}$ (m/s)	0.24
y (m)	0.27	$\dot{z}$ (m/s)	0.19
z (m)	0.11	$\theta(deg)$	0.31
$\dot{x}$ (m/s)	0.48	$\phi(deg)$	0.18

The UAV tends to show satisfactory responses for the spiral trajectory With the highest RMSE incurred at x position tracking. Regardless, this shows that the UAV would easily be able to follow the UAV through any trajectories that may require a large amount of rotations. Similar to the rectangular trajectory the UAV exhibits rapid movement in the beginning which is due to the controller adjusting to the aerodynamic and gyro disturbances present throughout the trajectory. Another observation made is regarding the RMSE for the velocity response in the x direction, this is due to the absence of sharp corners which tends to impose a jerk like response on the UAV.

We now move to an aggressive maneuver within a complex helical trajectory. Figures 5.14 to 5.18 show the UAV responses.

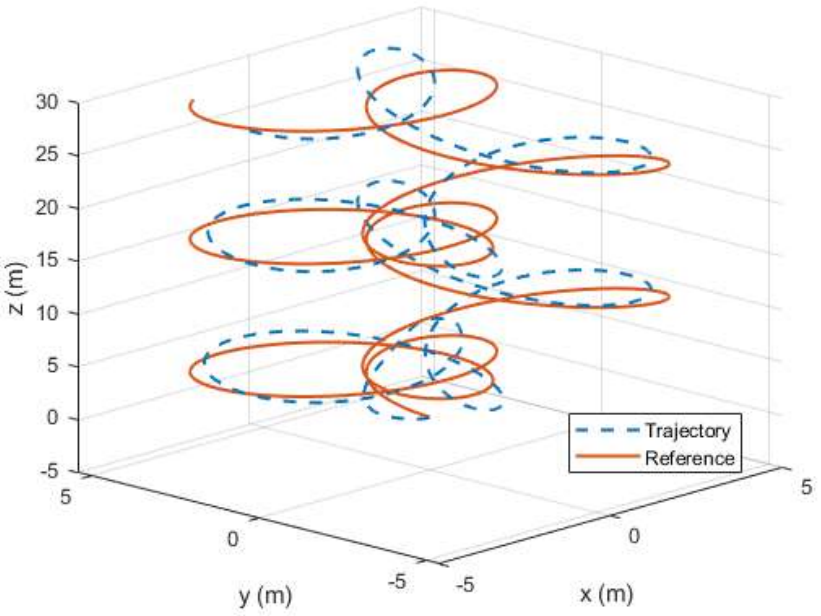


Figure 5.14: Complex Helical Trajectory

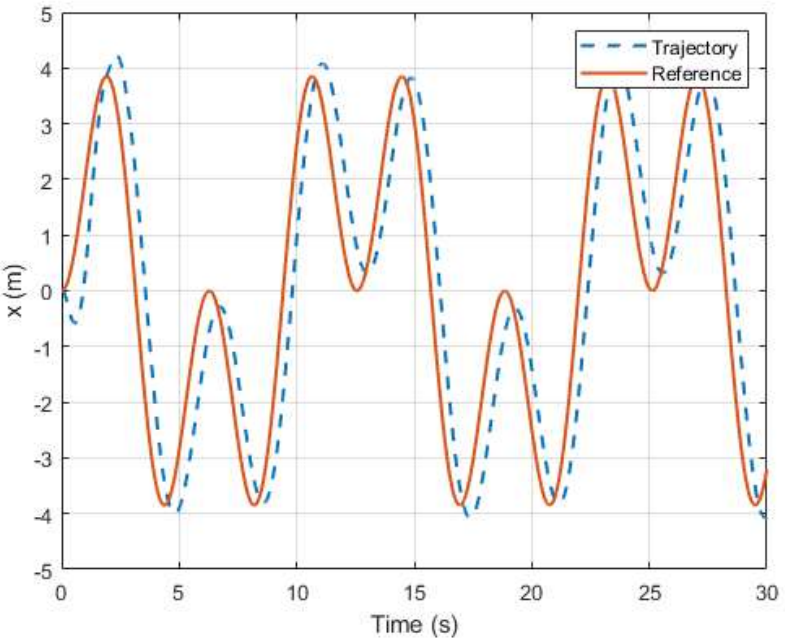


Figure 5.15: x Position Response in Helical Trajectory

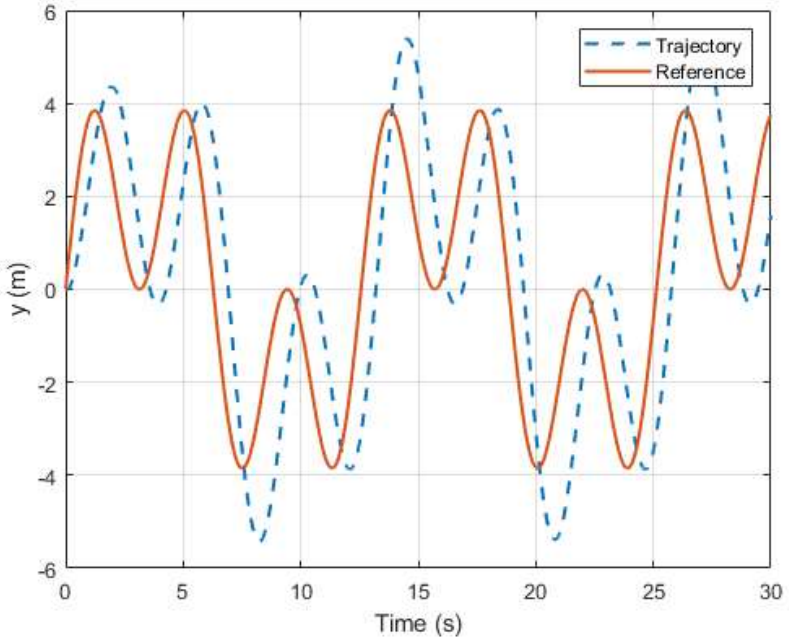
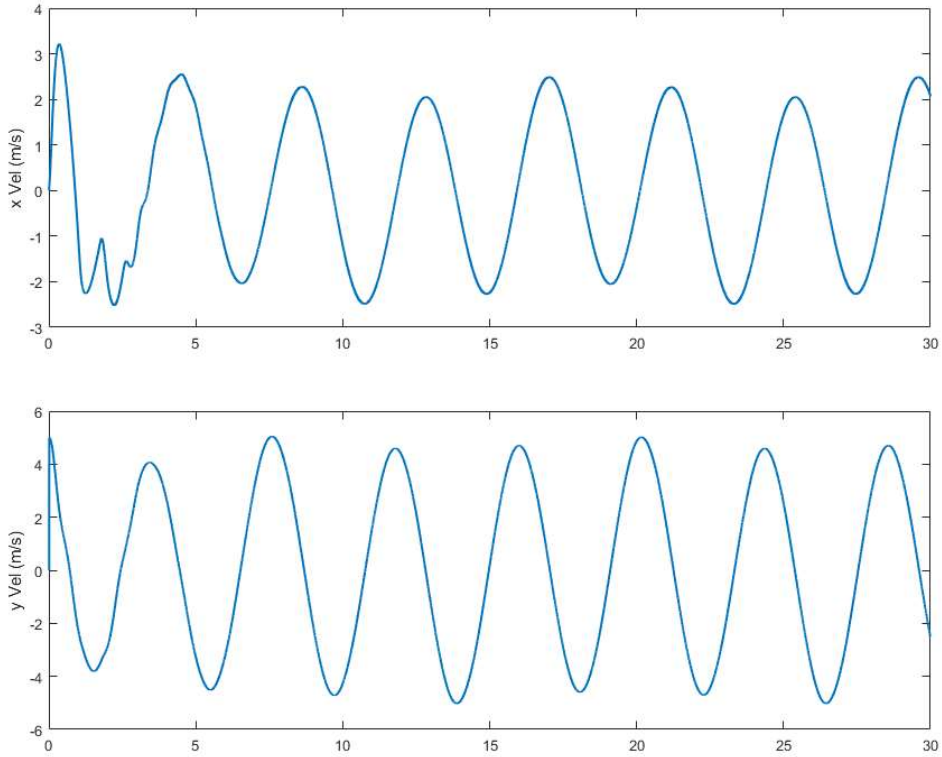


Figure 5.16: y Position Response in Helical Trajectory



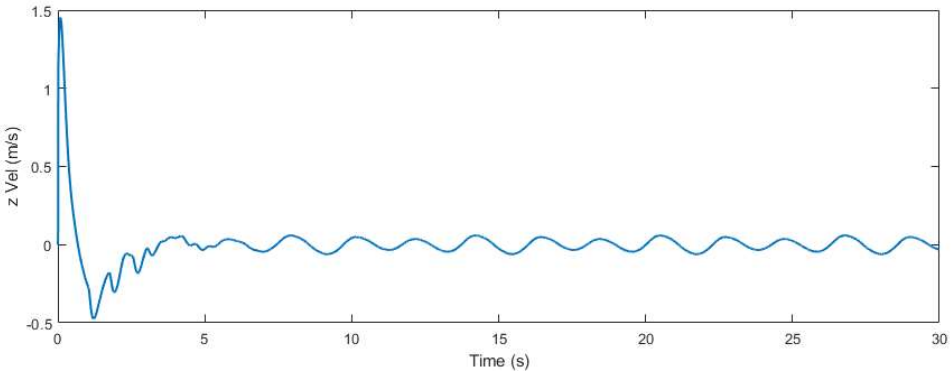


Figure 5.17: Velocity Errors in Complex Trajectory

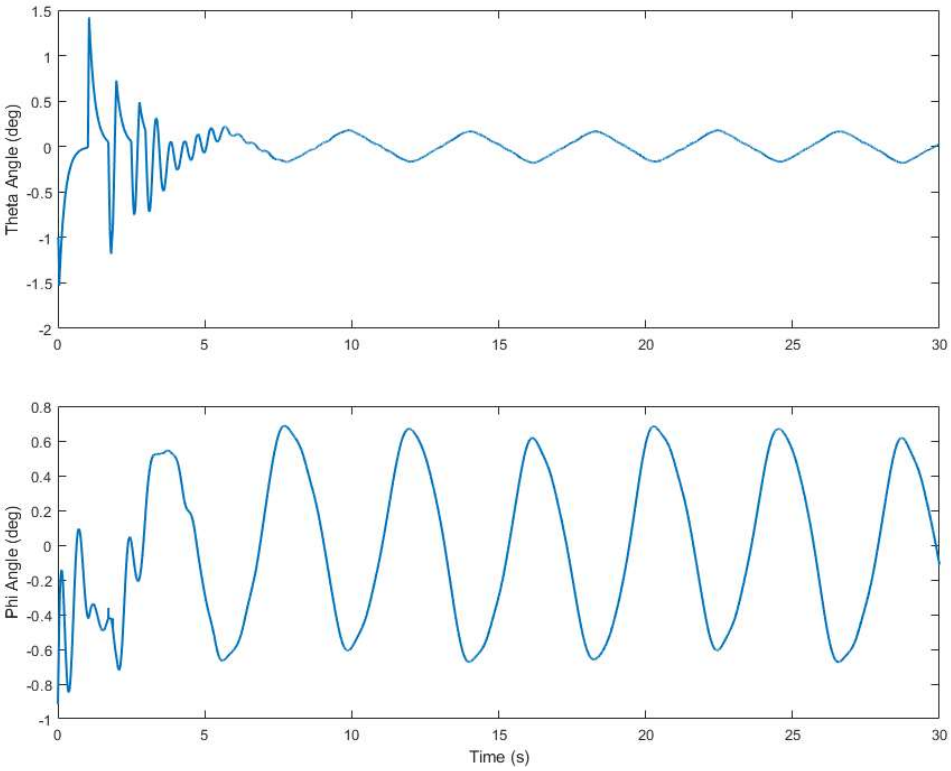


Figure 5.18: Angular Errors in Complex Trajectory

Table 5.4: Error Parameters Spiral Trajectory

Parameter	**RMS Error**	**Parameter**	**RMS Error**
x (m)	1.13	$\dot{y}$ (m/s)	3.21
y (m)	2.24	$\dot{z}$ (m/s)	0.15
z (m)	0.08	$\theta(deg)$	0.25
$\dot{x}$ (m/s)	1.70	$\phi(deg)$	0.45

The UAV evidently struggles to follow the trajectory, The controller fails to meet the demands of the maneuver and shows large errors on position and velocity re-

sponses. Regardless, from the ascending spiral responses and rectangular responses it is evident that UAV can easily hold its height and track the x and y position for non aggressive flights. We now design trajectories for generating synthetic data for tracking a UGV, within UAV performance limits.

5.2 UGV State Estimation

For the UGV state estimation we expect the camera on board the UAV to carry out two folds task. The first is to carry out visual odometry. For this purpose the UAV has to determine useful features from its image plane and compare them with subsequent frames to understand how its pose has changed from one frame to the next. Figure 5.19 shows a captured frame whist the UAV tracks the UGV while passing through waypoints in our simulation environment.

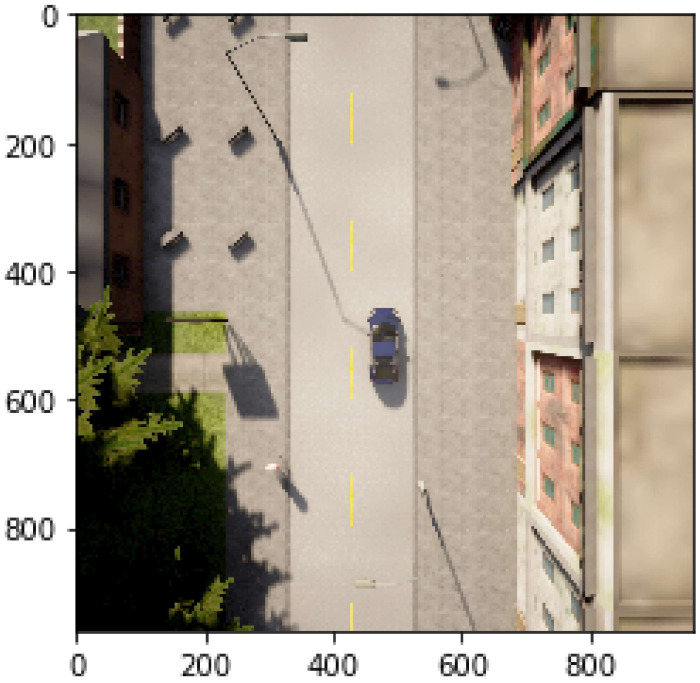


Figure 5.19: UAV Tracking UGV in Simulation Environment.

From the image we can see that a lot of texture is available from the given captured frame. To make meaningful assumptions we find features from another image a few seconds after this instance as shown in Figure 5.20.

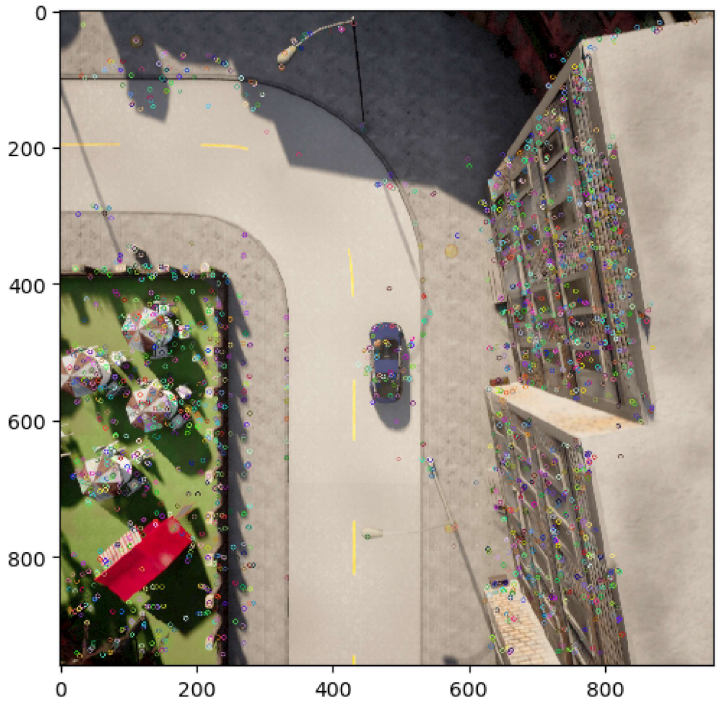


Figure 5.20: Extracted Features from Image.

However, an immediately noticeable problem is that much of the features that have been detected lie on our UGV. Since the UGV is mobile and is passing the same waypoints as the UAV this may result in incorrect odometry estimations. Hence, it is imperative to filter the features on top of our UGV. To do this we make use of the object detection network mentioned in Chapter 4. Through this algorithm we can understand the pixels that belong to the UGV and can seamlessly filter them out. The result is shown in Figure 5.21.

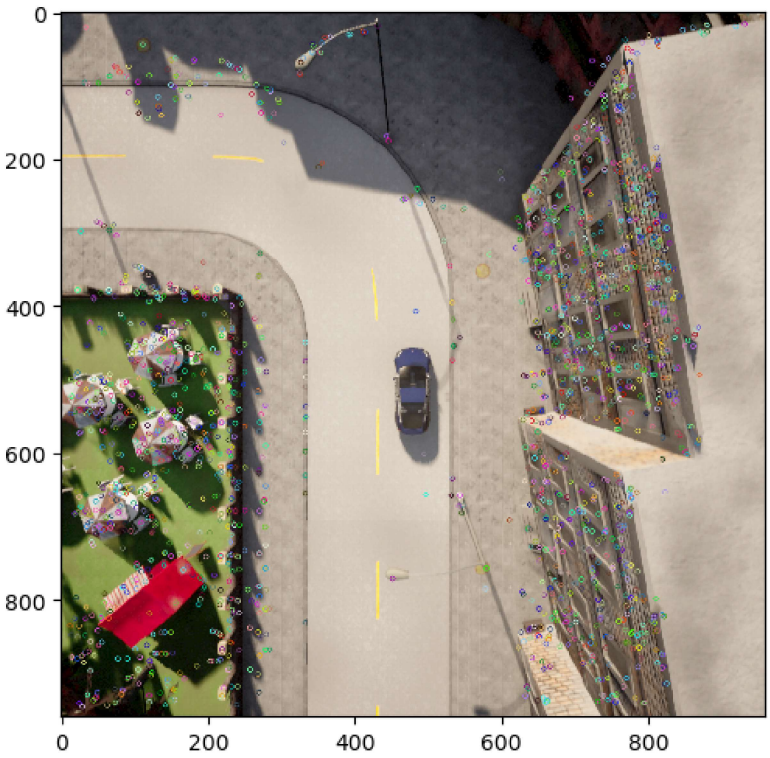


Figure 5.21: Filtered Features from Image.

From Figure 5.21 we can see that the UGV and consequently the UAV are both about to enter a turn, since feature points that may lead to erroneous results have been removed, the UAV carries out visual odometry to estimate its position in the local map over the turn.

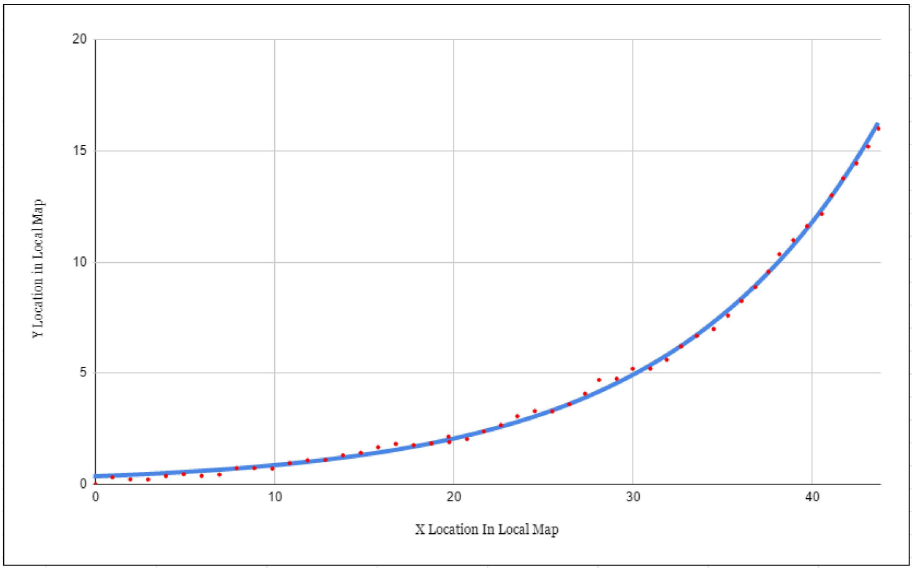


Figure 5.22: Visual Odometry Over a Turn.

The result given in Figure 5.22 shows that the estimates made by the camera correctly encapsulate the motion and current state of the UAV. It is to be noted that sice we are simulating a flying camera we do not need to transform from the camera axis to the body axis of the UAV.

Now that the UAV can establish its own position in the environment and can use the object detection network to establish the location of the UGV in its image frame, we can begin to extract useful information regarding the environment for control purposes. Taking Figure 5.19 as our base image one of the desirable information from the map is drive-able terrain. Since, we established a mathematical model for the UGV that can represents states as how far or close the UGV is from the lane markings, finding their positions on the image and consequently the world can be very beneficial. For this purpose we can make use of image segmentation that can break down the pixels in the image into various sections. Figure 5.23 shows how pixels can be alloted specific values to make this distinction through a color map.

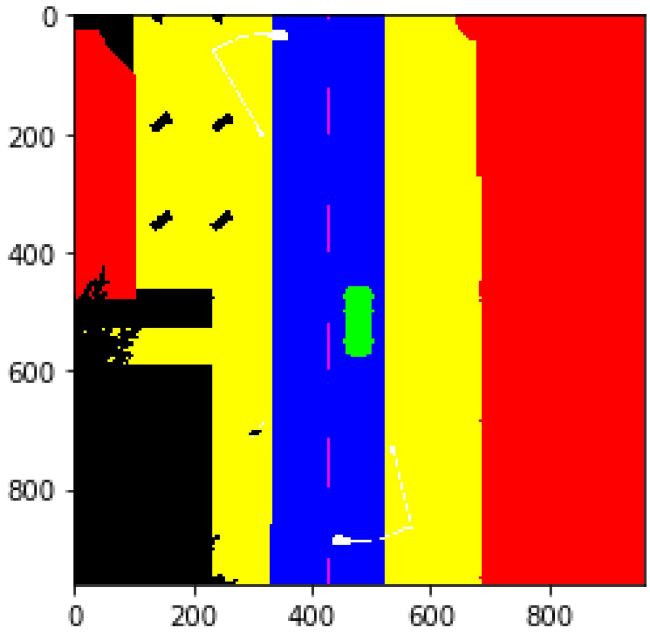


Figure 5.23: Semantically Segmented Image.

Using this information we can create a mask for the drive-able surface and extract

road information. We can further calculate hough lines about the masked area to find road markings which are of interest to us. Both of these steps are shown in Figure 5.24 and 5.25 below.

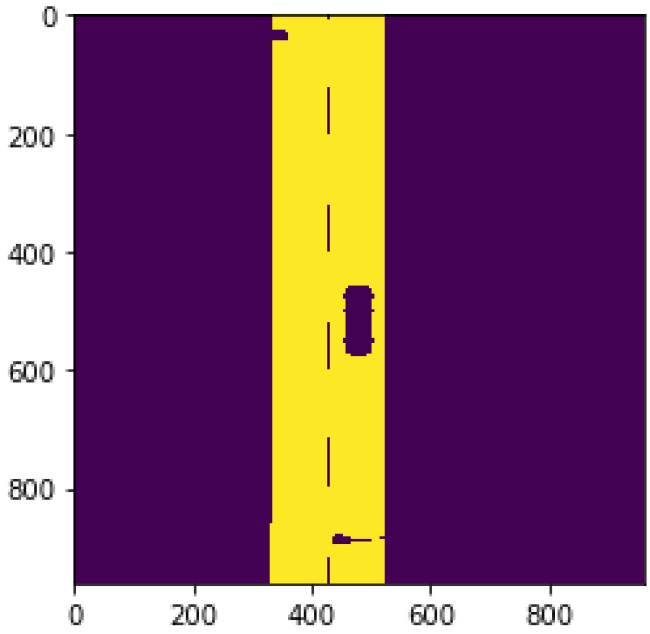


Figure 5.24: Masked Drive-able region extracted from image.

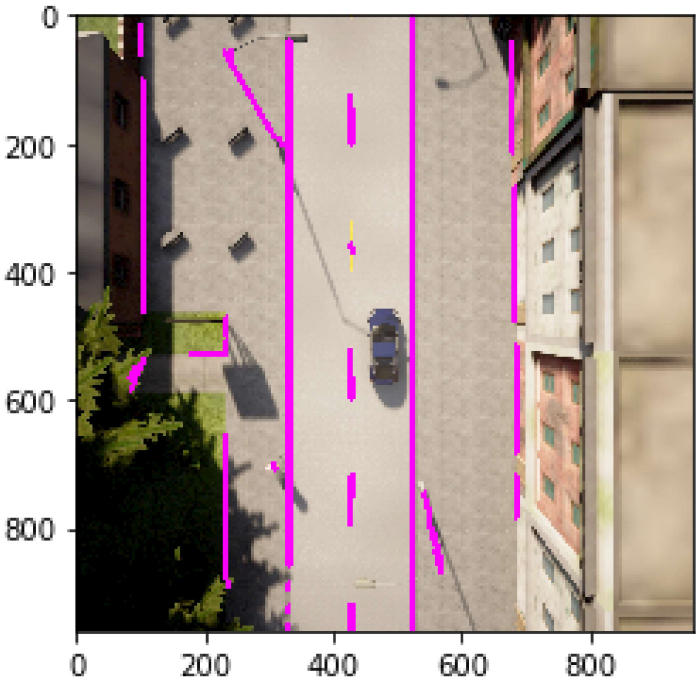


Figure 5.25: Hough Lines Extracted from Road.

In Figure 5.25 we have extracted hough lines from our image. For further precision

we can threshold the lines and find those that best fit the lane boundaries, however in the current frame we face on of the primal challenges of vision based sensors that pertain to occlusions and artifacts such as shadows in the images. On thresholding the algorithm incorrectly identifies the the shadows made by two street lights to be the road as they occupy a large portion of the image within the drive-able surface.

After performing these various steps, for the current image we are able to ascertain additional information regarding relevant pixels. The pixels pertaining to the UGV especially their centroid is of special interest to us since all estimates regarding the vehicles position will be based made on the centroid pixel location from the camera. For an aerial top down view, we expect the size of the bounding box on the UGV to remain the same as it is tracked through each frame, which can help reduce the variance of calculating different pixels around the UGV which may affect the estimates.

We now initialize the UGV under the UAV view as depicted in Figure 5.26.

Figure 5.26: UAV camera's aerial view.

In the given scenario the UAV uses an onboard camera and IMU to carry out mapping while passing through a pre set trajectory. The Velocity of the UGV is set such that it stays within the camera frame during the entire motion. The UAV is also equipped with an IMU which it uses to make a bias prone estimate of its states.

The IMU onboard the UGV runs at a faster frequency than the frequency of camera updates, when updates are received UGV uses a Kalman filter to rectify its states. The loop for the UGV runs as,

- UGV updates state with IMU readings.
- UGV propagates its uncertainty within the Kalman filter.
- If an update is available from the UAV, the UGV calculates the Kalman gain and computes error states.
- With the new update corrections to the predicted state are made.
- Finally covariance is corrected to complete the Kalman loop.

We develop synthetic data using our simulator and for a specific road scenario we test the affect of velocity of the vehicle on the accuracy of state estimation. We run the given scenario in Figure 5.27 for two different velocities and the same noise variance in the gyro and odometry.

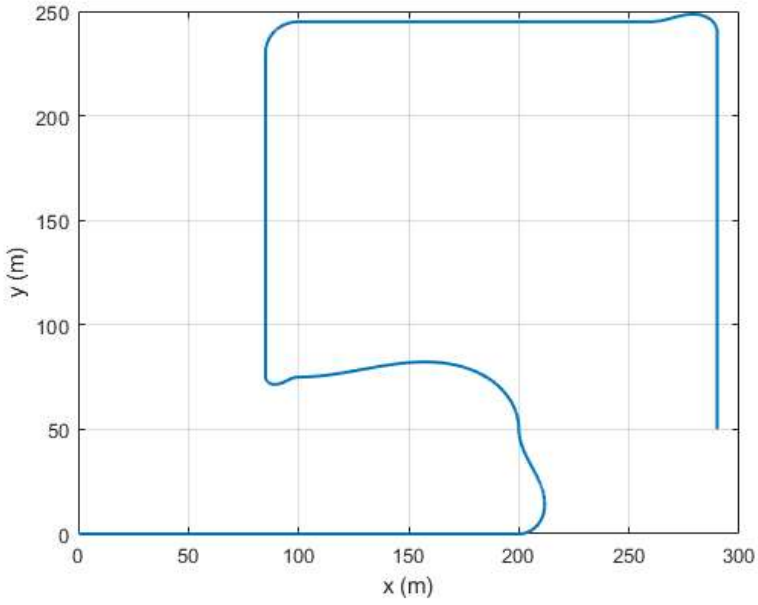


Figure 5.27: Ground Truth for Simulation

First the unmanned vehicles are tasked to travel through the waypoints at a reference speed of 20 m/s. Figure 5.28 to 5.30 show the ground vehicle estimates for this

scenario.

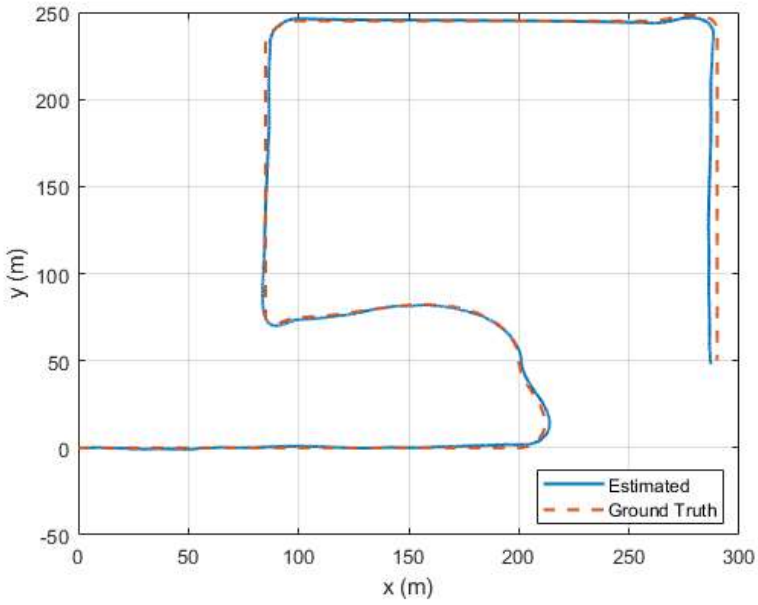


Figure 5.28: UGV State Estimation for 20 m/s

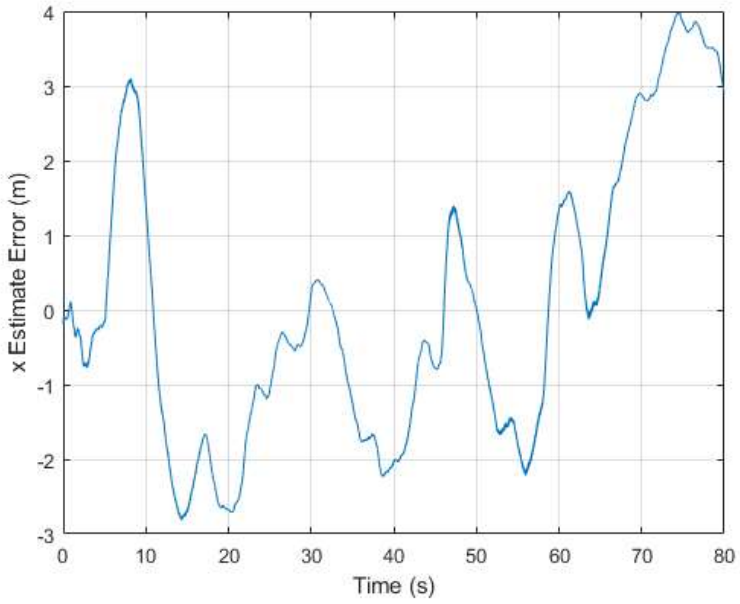


Figure 5.29: Error in x position

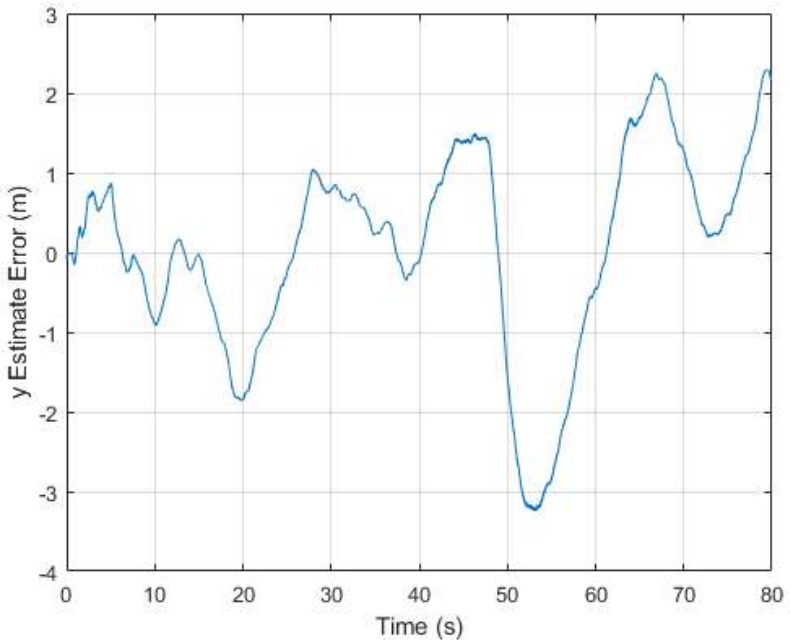


Figure 5.30: Error in y position

Table 5.5: UGV State Estimation Error Parameters at velocity 20 m/s

Parameter	**RMS Error**	**Max Error**
x (m)	1.90	3.97
y (m)	1.26	3.1

At 20 m/s we can measure the the state of UGV to a degree of error. With a max error of 3.97 m, the process is still liable of losing an accurate measure of position of the UGV however, it prevents the IMU from continually drifting and tends to lose accuracy at the high speed turns. Next from Figure 5.31 to 5.33 we check the viability of the system at twice the speed.

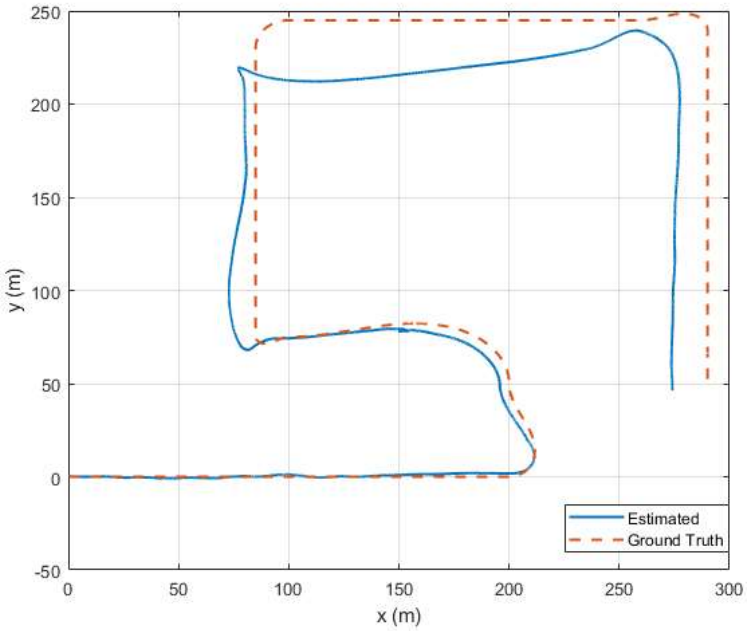


Figure 5.31: UGV State Estimation for 40 m/s

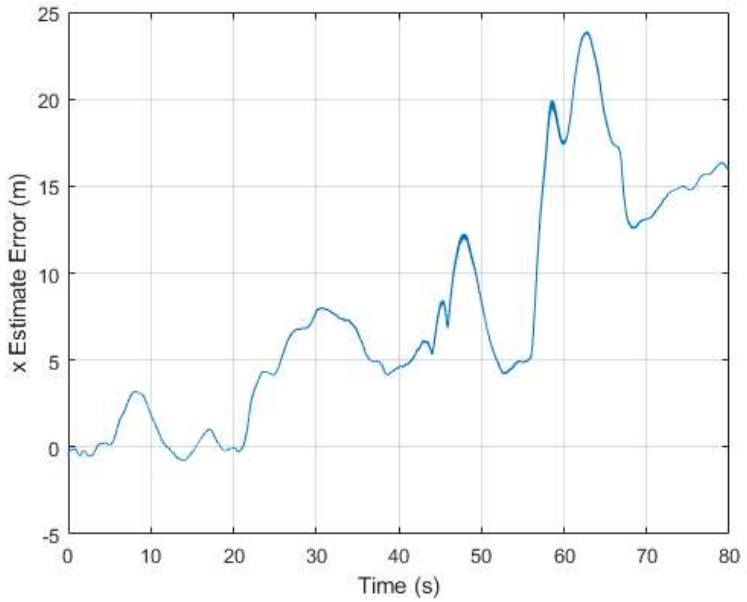


Figure 5.32: Error in x position

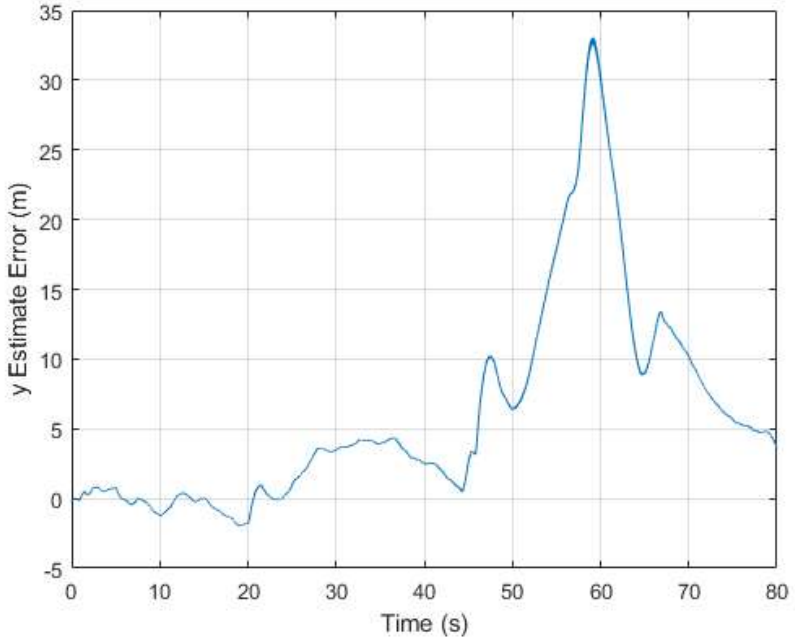


Figure 5.33: Error in y position

Table 5.6: UGV State Estimation Error Parameters at velocity 40 m/s

Parameter	**RMS Error**	**Max Error**
x (m)	10.5	23.9
y (m)	13.8	33.3

From these results, the system shows poor performance at a speed of 40 m/s. The system becomes reliant on the IMU as visual odometry fails at turns and misses important correspondences between frames.

Regardless, we find the capped speed for the system, and we run another simulation at a slower speed for the vehicles with inhibited noise on the odometry readings.

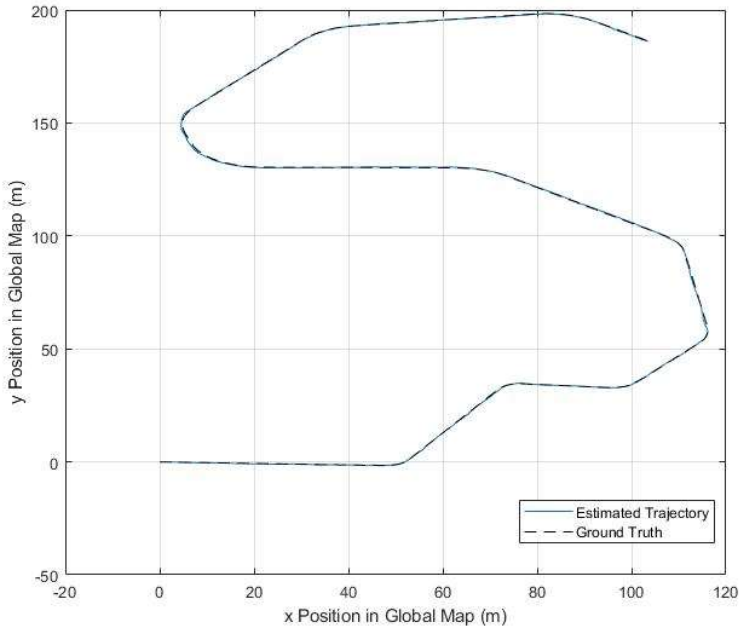


Figure 5.34: UGV Position Estimation

From Figure 5.34, we can see that the filtered estimate of the UGV closely resembles the ground truth trajectory that it is following with a mean error of 4.07 percent in the x,y plane. We specifically mention the x,y plane since for our estimation tasks we have used a simplified condition for the essential matrix assuming that the height of the UGV is known and remains fixed through out the mission. However, that is not the case as the IMU on board the UGV does collect information regarding the changing levels of the terrain. This results in noise being accumulated over the z-axis that the IMU struggles to filter alone. The sizeable noise is shown in Figure 5.35.

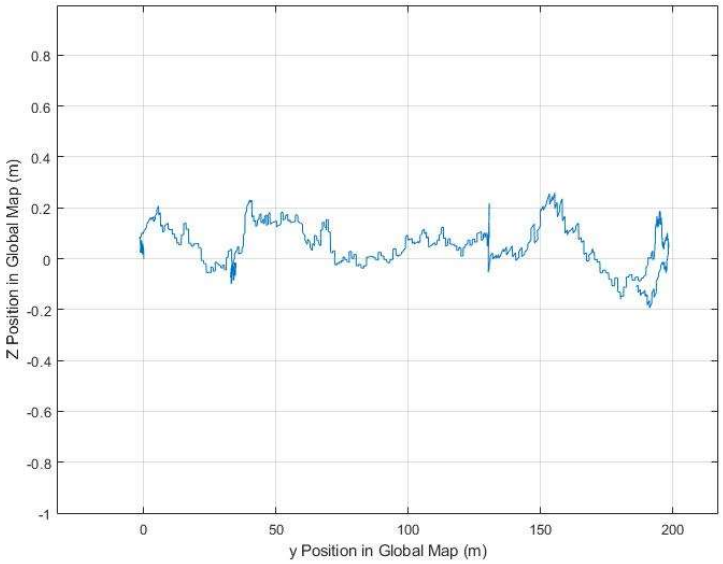


Figure 5.35: Noise Accumulated across the y-z Plane

The addition of a higher noise ratio in both IMUs tends to weaken the overall estimate of the UGV. However, the linear velocity of the vehicles tends to have an even greater impact on the performance. This may partly be due to the features between camera frames being incorrectly matched due to a larger base line difference between two subsequent camera images. Based on the dynamic model for the UGV defined we prepare a scenario in CARLA where the UGV is provided with with scheduled position and velocity estimates to match. The longitudinal dynamics are controlled by way of a standard PID controller while the lateral dynamics are matched through a Stanley Controller.

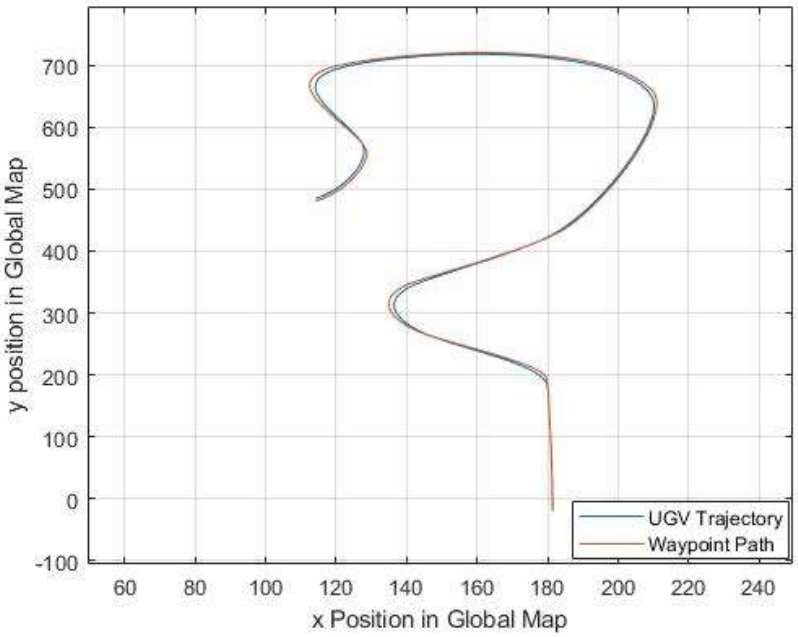


Figure 5.36: UGV Path Tracking.

Figure 5.36 shows that the UGV can be controlled to detect and follow the desired trajectory, the throttle of the UGV is capped at 20 m/s as to not prevent the estimation from giving erroneous state estimates.

Chapter 6

Conclusion and Future Work

In this thesis, we explored an alternate approach to estimating the position of a UGV in GPS denied areas using a camera in a UAV-UGV heterogeneous setting. The strategy involved providing the UAV with the ability to be able to create a map in which it was able to estimate the location of the UGV in the global frame. Taking advantage of the planar nature of the image scene, the projective space was isomorphised to its homography. This allowed us to understand the camera pose in successive frames with a certain degree of error, that was further corrected by using IMU readings in Kalman fusion. With satisfactory results on the UAV state estimation we moved to localize the UGV by deriving information from the image frames. To this end, segmentation was used to understand the drive able area in the scene and thresholded hough lines were used to extract the current lane markings. The remaining task involved understanding the location of the UGV in the global frame. Which was achieved using object detection to mark the pixels pertaining to them, since multiple pixels on the image frame are occupied by the car, the centroid position of the drawn bounding box is used for localizing the vehicle in order to maintain the pixel pertaining to the same point on the vehicle in successive frames. For constant altitude fights without any aggressive maneuvers or pure rotations, this approach tended to provide noisy estimates of the UGV position estimates, mainly due to the centroid approach not being very robust. Hence, an IMU sensor

on placed on the center of mass of the UGV was introduced. This IMU was used to for Kalman Fusion in order to improve state estimates of the UGV. Finally, a Stanley controller was used to test UGV by running the vehicle through predefined trajectory points. While, the UGV was able to follow the trajected path closely, the nature of the control system used was not able to cater for slower speeds however, provided reasonable results for higher speeds.

This work currently makes use of a simplified controller and standard Kalman filtering to accurately control and estimate the UGV; however the use of model predictive controller (MPC) along with an optimal approach to sensor fusion could drastically improve the results.

REFERENCES

[1] N. Michael, S. Shen, K. Mohta, *et al.*, "Collaborative mapping of an earthquake damaged building via ground and aerial robots," vol. 92, Jul. 2012, ISBN: 978-3-642-40685-0. DOI: 10.1007/978-3-642-40686-7_3.

[2] B. Kim, M. Kaess, L. Fletcher, *et al.*, "Multiple relative pose graphs for robust cooperative mapping," in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 3185–3192. DOI: 10.1109/ROBOT.2010.5509154.

[3] A. Gawel, R. Dube, H. Surmann, J. Nieto, R. Siegwart, and C. Cadena, "3d registration of aerial and ground robots for disaster response: An evaluation of features, descriptors, and transformation estimation," Oct. 2017, pp. 27–34. DOI: 10.1109/SSRR.2017.8088136.

[4] R. Mahony, R. Beard, and V. Kumar, "Modeling and control of aerial robots," pp. 1307–1334, Jan. 2016. DOI: 10.1007/978-3-319-32552-1_52.

[5] S. Minaeian, J. Liu, and Y.-J. Son, "Vision-based target detection and localization via a team of cooperative uav and ugvs," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, pp. 1–12, Jan. 2015. DOI: 10.1109/TSMC.2015.2491878.

[6] M. Tjahjadi and F. Agustina, "Single image orientation of uav's imagery using orthogonal projection model," Nov. 2017, pp. 18–23. DOI: 10.1109/ISYG.2017.8280668.

[7] E. Mueggler, M. Faessler, F. Fontana, and D. Scaramuzza, "Aerial-guided navigation of a ground robot among movable obstacles," Oct. 2014. DOI: 10.1109/SSRR.2014.7017662.

[8] X. Zheng, S. Galland, X. Tu, Q. Yang, A. Lombard, and N. Gaud, "Obstacle avoidance model for uavs with joint target based on multi-strategies and follow-up vector field," *Procedia Computer Science*, vol. 170, pp. 257–264, 2020, The 11th International Conference on Ambient Systems, Networks and Technologies (ANT) / The 3rd International Conference on Emerging Data and Industry 4.0 (EDI40) / Affiliated Workshops, ISSN: 1877-0509.

[9] M. Masný, K. Weis, and M. Biskupic, "Application of fixed-wing uav-based photogrammetry data for snow depth mapping in alpine conditions," *Drones*, vol. 5, p. 114, Oct. 2021. DOI: 10.3390/drones5040114.

[10] C. Pfeifer, A. Barbosa, O. Mustafa, H.-U. Peter, A. Brenning, and M.-C. Rümmler, "Using fixed-wing uav for detecting and mapping the distribution and abundance of penguins on the south shetlands islands, antarctica," *Drones*, vol. 3, p. 39, Apr. 2019. DOI: 10.3390/drones3020039.

[11] X. Wang, H. Zhu, D. Zhang, D. Zhou, and X. Wang, "Vision-based detection and tracking of a mobile ground target using a fixed-wing uav," *International Journal of Advanced Robotic Systems*, vol. 11, pp. 1–11, Sep. 2014. DOI: 10.5772/58989.

[12] K. Waldron and R. McGhee, "The adaptive suspension vehicle," *IEEE Control Systems Magazine*, vol. 6, no. 6, pp. 7–12, 1986. DOI: 10.1109/MCS.1986.1105145.

[13] X. yu, C. Fu, and K. Chen, "Modeling and control of a single-legged robot," *Procedia Engineering*, vol. 24, pp. 788–792, Dec. 2011. DOI: 10.1016/j.proeng.2011.11.2738.

[14] Q. Ruan, J. Wu, and Y.-a. Yao, "Design and analysis of a multi-legged robot with pitch adjustive units," *Chinese Journal of Mechanical Engineering*, vol. 34, Dec. 2021. DOI: 10.1186/s10033-021-00578-z.

[15] M. L. Okyen, "Biomimetic bi-pedal humanoid: Design, actuation, and control implementation with focus on robotic legs," 2013.

[16] "Modeling, stability and control of biped robots—a general framework," *Automatica*, vol. 40, no. 10, pp. 1647–1664, 2004, ISSN: 0005-1098. DOI: https://doi.org/10.1016/j.automatica.2004.01.031.

[17] R. W. Sinnet and A. D. Ames, "Bio-inspired feedback control of three-dimensional humanlike bipedal robots," *Journal of Robotics and Mechatronics*, vol. 24, no. 4, p. 595, 2012.

[18] N. Yadav, T. Raina, N. Gupta, and N. V. Shrivas, "Design and development of a structurally stable quadruped robot for surveillance," in *AIP Conference Proceedings*, AIP Publishing LLC, vol. 2148, 2019, p. 030 028.

[19] Z. Youcef, C. Bensaci, and D. Pomorski, "Nonlinear control of a differential wheeled mobile robot in real time - turtlebot 2," Dec. 2018.

[20] R. H. Abiyev, I. S. Günsel, N. Akkaya, E. Aytac, A. Çağman, and S. Abizada, "Fuzzy control of omnidirectional robot," *Procedia Computer Science*, vol. 120, pp. 608–616, 2017, 9th International Conference on Theory and Application of Soft Computing, Computing with Words and Perception, ICSCCW 2017, 22-23 August 2017, Budapest, Hungary, ISSN: 1877-0509.

[21] E. Mingachev, R. Lavrenov, E. Magid, and M. Svinin, "Comparative analysis of monocular slam algorithms using tum and euroc benchmarks," in Sep. 2020, pp. 343–355, ISBN: 978-981-15-5579-4. DOI: 10.1007/978-981-15-5580-0_28.

[22] S. Li, Z. Yan, H. Li, and K.-T. Cheng, *Exploring intermediate representation for monocular vehicle pose estimation*, 2021. arXiv: 2011.08464.

[23] J. Renoux, A.-I. Mouaddib, and S. L. Gloannec, "A decision-theoretic planning approach for multi-robot exploration and event search," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 5287–5293. DOI: 10.1109/IROS.2015.7354123.

[24] A. Ravankar, A. Ravankar, T. Emaru, and Y. Kobayashi, "Visual-aided multi-robot mapping and navigation using topological features," vol. 42, Nov. 2019, p. 6580. DOI: 10.3390/ecsa-6-06580.

[25] R. Kaslin, P. Fankhauser, E. Stumm, *et al.*, "Collaborative localization of aerial and ground robots through elevation maps," Oct. 2016, pp. 284–290. DOI: 10.1109/SSRR.2016.7784317.

[26] X. Fang, Y. Wang, and Q. Lin, "Flight control of convertible uav based on extended-state-observer," in *2011 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC)*, 2011, pp. 369–373. DOI: 10.1109/AIMSEC.2011.6010394.

[27] M. N. Soorki, H. A. Talebi, and S. K. Y. Nikravesh, "A leader-following formation control of multiple mobile robots with active obstacle avoidance," in *2011 19th Iranian Conference on Electrical Engineering*, 2011, pp. 1–6.

[28] M. Turpin, N. Michael, and V. Kumar, "Capt: Concurrent assignment and planning of trajectories for multiple robots," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 98–112, 2014. DOI: 10.1177/0278364913515307. [Online]. Available: https://doi.org/10.1177/0278364913515307.

[29] A. Khaleghi, D. Xu, S. Minaeian, *et al.*, "A comparative study of control architectures in uav/ugv-based surveillance system," pp. 3455–3464, Jan. 2014.

[30] E. Mueggler, M. Faessler, F. Fontana, and D. Scaramuzza, "Aerial-guided navigation of a ground robot among movable obstacles," in *2014 IEEE International Symposium on Safety, Security, and Rescue Robotics (2014)*, 2014, pp. 1–8. DOI: 10.1109/SSRR.2014.7017662.

[31] M. Garzón, J. Valente, D. Zapata, and A. Barrientos, "An aerial-ground robotic system for navigation and obstacle mapping in large outdoor areas," *Sensors*, vol. 13, no. 1, pp. 1247–1267, 2013, ISSN: 1424-8220. DOI: 10.3390/s130101247.

[32] L. Matignon, L. Jeanpierre, and A.-i. Mouaddib, "Decentralized multi-robot planning to explore and perceive," *Acta Polytechnica*, vol. 55, p. 169, Jun. 2015. DOI: 10.14311/AP.2015.55.0169.

[33] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[34] A. Sidea, R. Brogaard, N. Andersen, and O. Ravn, "General model and control of an n rotor helicopter," *Journal of Physics: Conference Series*, vol. 570, p. 052 004, Dec. 2014. DOI: 10.1088/1742-6596/570/5/052004.

[35] S. A. A. Hosseini, M. Zamanian, S. Shams, and A. Shooshtari, "Vibration analysis of geometrically nonlinear spinning beams," *Mechanism and Machine Theory*, vol. 78, pp. 15–35, Aug. 2014. DOI: 10.1016/j.mechmachtheory.2014.02.015.

[36] D. Wang and F. Qi, "Trajectory planning for a four-wheel-steering vehicle," vol. 4, Feb. 2001, 3320–3325 vol.4, ISBN: 0-7803-6576-3. DOI: 10.1109/ROBOT.2001.933130.

[37] R. Rajamani, "Lateral vehicle dynamics," in Oct. 2012, pp. 15–46, ISBN: 978-1-4614-1432-2. DOI: 10.1007/978-1-4614-1433-9_2.

[38] R. A. White and H. H. Korst, "The determination of vehicle drag contributions from coast-down tests," *SAE Transactions*, vol. 81, pp. 354–359, 1972, ISSN: 0096736X, 25771531.

[39] S.-J. Hwang, J.-S. Chen, L. Liu, and C.-C. Ling, "Modeling and simulation of a powertrain-vehicle system with automatic transmission," *International*

Journal of Vehicle Design - INT J VEH DES, vol. 23, Jan. 2000. DOI: 10.1504/IJVD.2000.001888.

[40] D. de Bruin and P. van den Bosch, "Lateral control of a four-wheel steered vehicle," *IFAC Proceedings Volumes*, vol. 31, no. 2, pp. 1–5, 1998, IFAC Workshop on Intelligent Components for Vehicles (ICV'98), Seville, Spain, 23-24 March, ISSN: 1474-6670. DOI: https://doi.org/10.1016/S1474-6670(17)44163-2.

[41] J. GULDNER, H.-S. TAN, and S. PATWARDHAN, "Analysis of automatic steering control for highway vehicles with look-down lateral reference systems," *Vehicle System Dynamics*, vol. 26, no. 4, pp. 243–269, 1996. DOI: 10.1080/00423119608969311.

[42] A. H. Al-Kaff, *Vision-based navigation system for unmanned aerial vehicles*, 2017.

[43] C. G. Harris and M. J. Stephens, "A combined corner and edge detector," in *Alvey Vision Conference*, 1988.

[44] T. Lindeberg, "Scale invariant feature transform," in May 2012, vol. 7. DOI: 10.4249/scholarpedia.10491.

[45] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," vol. 3951, Jul. 2006, pp. 404–417, ISBN: 978-3-540-33832-1. DOI: 10.1007/11744023_32.

[46] S. Ehsan, A. Clark, N. Rehman, and K. McDonald-Maier, "Integral images: Efficient algorithms for their computation and storage in resource-constrained embedded vision systems," *Sensors*, vol. 15, pp. 16 804–16 830, Jul. 2015. DOI: 10.3390/s150716804.

[47] D. Viswanathan, "Features from accelerated segment test (fast)," 2011.

[48] M. Flannery, S. Fenn, and D. Budden, *Ransac: Identification of higher-order geometric features and applications in humanoid robot soccer*, 2013. arXiv: 1310.5781 [cs.RO].

[49] B. Li, L. Heng, G. H. Lee, and M. Pollefeys, "A 4-point algorithm for relative pose estimation of a calibrated camera with a known relative rotation angle," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 1595–1601. DOI: 10.1109/IROS.2013.6696562.

[50] J. Redmon and A. Farhadi, *Yolov3: An incremental improvement*, 2018.

[51] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788. DOI: 10.1109/CVPR.2016.91.

[52] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6517–6525. DOI: 10.1109/CVPR.2017.690.

[53] A. K. Burusa, "Visual-inertial odometry for autonomous ground vehicles," 2017.

[54] H. Fahraeus, *Fusion of imu and monocular-slam in a loosely coupled ekf*, 2017.