University of Cincinnati

Date: 2/28/2019

I, **Srijanee Biswas**, hereby submit this original work as part of the requirements for the degree of Master of Science in Aerospace Engineering.

It is entitled:

Goal-Aware Robocentric Mapping and Navigation of a Quadrotor Unmanned Aerial Vehicle

Student's name: **Srijanee Biswas**

This work and its defense approved by:

Committee chair: Rajnikant Sharma, Ph.D.

Committee member: Manish Kumar, Ph.D.

Committee member: Ou Ma, Ph.D.

32945

Goal-Aware Robocentric Mapping and Navigation of a Quadrotor Unmanned Aerial Vehicle

A thesis submitted to the
Graduate School
of the University of Cincinnati
in partial fulfillment of the
requirements for the degree of

Master of Science
in the Department of Aerospace Engineering and Engineering Mechanics
of the College of Engineering and Applied Sciences

by

Srijanee Biswas

B.Tech, Maulana Abul Kalam Azad University of Technology, India
July 2015

Committee Chair: R. Sharma, Ph.D.

UNIVERSITY OF CINCINNATI

Abstract

Department of Aerospace Engineering and Engineering Mechanics
College of Engineering and Applied Sciences

Master of Science

by Srijanee Biswas

Autonomous vehicles have become a reality in many military and civilian applications. The ability to deploy them in constrained environments, such as regions with limited or no Global Positioning System (GPS) access, or no a-priori map information, can not only further their application space but also add capability to successfully complete tasks that are otherwise considered dull, dirty or dangerous for humans. This Thesis proposes and implements an autonomous navigation solution for an Unmanned Aerial Vehicle (UAV) using a robot-centered reference frame. An Extended Kalman Filter (EKF) is used to estimate the relative orientation of the UAV with respect to an user-defined goal and other objects of interest in the environment (called landmarks). A visual tracker continuously tracks these objects and based on the camera parameters, calculates their bearing measurement with respect to the UAV. This method uses a bearing-only measurement model to update the states of the system. The goal is selected real-time from the video feed provided by the UAV's onboard camera and the UAV has to navigate to it while avoiding obstacles along its path. A combined PN-guidance and obstacle avoidance controller is used for this purpose. A detailed 2D observability analysis is performed to find the sufficient conditions required for the system to be observable. The problem formulation is corroborated through extensive simulation and hardware results.

Copyright 2019, Srijanee Biswas

This document is copyrighted material. Under copyright law, no parts of this document may be reproduced without the expressed permission of the author.

“The more that you read, the more things you’ll know. The more that you learn, the more places you’ll go.”

- Dr. Seuss

Acknowledgements

To my family and friends, thank you for encouraging me on all my endeavours and inspiring me to follow my dreams. I know it must have been difficult to see me leave the nest, especially to settle on the opposite corner of the planet from you. Yet, you have supported me unconditionally throughout this process. I am thankful to my grandma, Manashi Chakraborty, for always lighting me up with that million dollar smile of hers. It helped me get through some of the difficult parts of this journey. Thank you to my parents, Goutam and Mousumi Biswas, for supporting me financially, for always believing in me and teaching me to appreciate the simple things in life.

To my advisor, Dr Rajnikant Sharma, thank you for sharing your vision and ideas for this project. I am thankful for your technical guidance and all the one-to-one meetings/discussion sessions. They helped me understand and solve some of the major technical roadblocks I faced along this project. Thank you for seeking our weekly research updates, they helped me maintain my research timeline and keep me on top of my progress. Thank you for letting me TA your class for two semesters. I really love sharing what I've learned and helping students understand these concepts in a lucid manner. I am also grateful for the opportunity to transfer to University of Cincinnati along with you. The prospect of changing my research major or looking for another advisor to fund my work was unnerving as a new graduate student.

To my first graduate school, Utah State University, thank you for accepting me and providing me an amazing environment to study in. It will always hold a special place in my heart. I'm grateful for having the opportunity to spend a semester there and learn from some ingenious professors. To my graduate school, University of Cincinnati, thank you for giving me the opportunity to learn from and collaborate with some amazing professors. Thank you Dr Manish Kumar and Dr Ou Ma for agreeing to be a part of my Thesis Committee. Thank you Dr Ali Minai for guiding our work on Search Area Optimization using multiple UAVs. Thank you, Bryan Brown, for always helping us sort out networking issues in the Motion Capture room.

To my lab mates, Anusna Chakraborty and Sohum Misra, thank you for the brainstorming sessions whenever we were stuck with our research problems. These helped me find solutions and understand concepts from our broader research topic in general. Thank you Raven, Seth, Eshaan, Tarun, Rohith and Melody for all the times we exchanged research ideas. I thank you all for making RISC lab a fun and constructive place to work in. I wish you all the success in your career.

To my teammates at Toyota Industrial Equipment Manufacturing (TIEM), thank you for the invaluable internship experience last summer and for being flexible enough to

accommodate my graduation timeline. I can't wait to be back there and work on newer, exciting projects.

Last but not the least, I'm thankful for engaging in pep talks with myself and for persevering through small and large setbacks alike. Having no prior background in UAVs to successfully implementing an autonomous navigation algorithm on a quadcopter, I've come a long way. I wish to keep challenging myself and learning more along the journey.

Contents

Abstract	i
Acknowledgements	iv
List of Figures	viii
List of Tables	xi
Abbreviations	xii
Nomenclature	xiii
1 Introduction	1
1.1 Motivation	1
1.2 Problem Description	3
1.3 Literature Review	4
1.3.1 Mapping and Localization	4
1.3.2 Path Planning	6
1.3.3 Visual Tracking	6
1.4 Summary of Contributions	7
1.5 Thesis Layout	8
2 Problem Methodology	9
2.1 State Definitions	9
2.2 State Estimation	10
2.2.1 Motion Model	12
2.2.2 Motion Model Jacobian	15
2.2.3 Measurement Model and Measurement Jacobian	17
2.3 Path Planning	19
2.3.1 Navigation to Goal	19
2.3.2 Obstacle Avoidance Controller	21
2.3.3 Velocity Controller	23
2.4 Visual Tracking	26
2.4.1 Goal Selection	26
2.4.2 Goal Tracking	28

2.4.2.1	Template Drift	29
2.4.2.2	Goal Reacquisition	30
2.4.3	Landmark Detection	31
2.4.4	Landmark Tracking	33
2.4.4.1	Data Association	34
2.5	Summary	36
3	Observability Analysis	37
3.1	Lie Derivatives	39
3.2	Case I: Goal Observability	41
3.3	Case II: Goal and Landmark Observability	43
3.4	Case III: Landmark-only Observability	46
3.5	Summary	50
4	System Overview	51
4.1	Robot platform	51
4.2	Sensors	53
4.3	Computer	53
4.4	Robot Operating System (ROS)	53
4.4.1	Tracker Node	54
4.4.2	Estimation Node	55
4.4.3	Controller Node	55
4.5	Test Facility and Setup	56
4.6	Simulation	57
4.7	Summary	58
5	Results	59
5.1	Simulation Results	59
5.1.1	Case I - Goal never leaves camera's FOV	60
5.1.2	Case II - Goal leaves camera's FOV (Single Obstacle Avoidance)	67
5.1.3	Case III - Multiple Obstacle Avoidance	70
5.2	Hardware Results	73
5.2.1	Case I: No Obstacle	75
5.2.2	Case II: Single Obstacle	80
5.3	Summary	84
6	Conclusions and Future Work	85
	Bibliography	87
	Experimental Challenges	97
	Linear Quadratic Regulator (LQR)	99
	Declaration of Authorship	101

List of Figures

1.1	(a) The blue dot shows the belief regarding the device's or car's current GPS location and the circle surrounding it depicts the uncertainty associated with this belief, (b) Incorrect GPS measurement - actual location is different than the current location provided by GPS. This may be caused by Multipath Errors, Line-of-Sight (LOS) blockage, etc.	2
2.1	State Modeling	9
2.2	UAV-Goal Engagement Geometry	20
2.3	A conceptual view of the UAV approaching an obstacle enroute to its goal	22
2.4	Definition of axes	24
2.5	Goal Template Selection	27
2.6	Predetermined template size cannot account for different goal sizes	27
2.7	Template based on goal size	28
2.8	Template size increases as UAV approaches the goal	29
2.9	Template size increases as UAV approaches the goal	30
2.10	Obstacle in the UAV's Direct Line-of-Sight to Goal	31
2.11	Goal leaves FOV while UAV tries to avoid obstacle	31
2.12	Landmarks and their surrounding templates	32
2.13	Pairwise distance calculation between existing LMs and new corners . . .	33
2.14	Landmark Association over multiple image frames	35
2.15	Incorrect Landmark Association	35
3.1	Cases when a system may become unobservable- (a) Circular Trajectory with range-only measurement, (b) Straight-line path with bearing-only measurement	37
3.2	Case I - Initial position of UAV and goal	42
3.3	Case I - $3 - \sigma$ Plots - α_g and d_g	42
3.4	Case II - Initial position of UAV, goal and LM_1	45
3.5	Case II - $3 - \sigma$ Plots - α_g and d_g	45
3.6	Case III - Initial position of UAV, goal and two LMs	48
3.7	Case III - $3 - \sigma$ Plots - α_g and d_g	48
4.1	System Overview	51
4.2	AR.Drone 2.0 used for experimental testing of this work	52
4.3	Direction of rotation of four rotors of a quadrotor UAV	52
4.4	Custom ROS message for publishing bearing measurements	54
4.5	Custom ROS message for publishing estimated states and their covariance	55
4.6	Indoor Motion Capture room	56
4.7	Simulation Block Diagram	57

5.1	Simulink Block Diagram	60
5.2	Case I - Initial Conditions. The UAV has bearing measurements from the goal and 5 LMs.	60
5.3	Case I - 2D view of the UAV's trajectory	62
5.4	True Vs Estimated Goal States - α_g , β_g and d_g	63
5.5	3 - σ Plots - α_g , β_g and d_g	63
5.6	True Vs Estimated LM3 States - θ_3 , η_3 and d_3	64
5.7	3 - σ Plots - θ_3 , η_3 and d_3	64
5.8	True Vs Estimated LM2 States - θ_2 , η_2 and d_2	65
5.9	3 - σ Plots - θ_2 , η_2 and d_2	65
5.10	Case I - Commanded Vs Actual Forward Velocity (m/s)	66
5.11	Case I - Commanded Vs Actual Yaw Rate (rad/s)	66
5.12	Case II - Initial Conditions	67
5.13	No measurement received from Goal	68
5.14	Case II - 2D view of the UAV's Trajectory	68
5.15	True Vs Estimated Goal States - α_g , β_g and d_g	69
5.16	3 - σ Plots - α_g , β_g and d_g	69
5.17	Case II - Commanded Vs Actual Forward Velocity (m/s)	70
5.18	Case II - Commanded Vs Actual Yaw Rate (rad/s)	70
5.19	Case III - 2D view of the UAV's Trajectory	71
5.20	True Vs Estimated Goal States - α_g , β_g and d_g	71
5.21	3 - σ Plots - α_g , β_g and d_g	72
5.22	Case III - Commanded Vs Actual Forward Velocity (m/s)	72
5.23	Case III - Commanded Vs Actual Yaw Rate (rad/s)	73
5.24	Initial position of UAV before take-off. It is currently tracked by 9 infrared cameras.	74
5.25	Goal position currently being tracked by 3 infrared cameras.	74
5.26	Goal Tracking with No Obstacle	75
5.27	Template as selected at the beginning of the mission	76
5.28	No Obstacle - 3D Trajectory Plot	76
5.29	No Obstacle - 2D Trajectory Plot	76
5.30	True versus Estimated Plots (No Obstacle) - α_g , β_g , d_g	77
5.31	3 - σ Plots (No Obstacle) - α_g , β_g , d_g	77
5.32	True versus Estimated Plots (No Obstacle) - θ_1 , η_1 , d_1	78
5.33	3 - σ Plots (No Obstacle) - θ_1 , η_1 , d_1	78
5.34	No Obstacle - Range Convergence Plot	79
5.35	No Obstacle - Commanded vs Actual Forward Velocity (m/s)	79
5.36	No Obstacle - Commanded vs Actual Yaw Rate (rad/s)	79
5.37	Goal Tracking during Single Obstacle Avoidance	80
5.38	Template as selected at the beginning of the mission	80
5.39	Single Obstacle - 3D Trajectory Plot	81
5.40	Single Obstacle - 2D Trajectory Plot	81
5.41	True versus Estimated Plots (Single Obstacle) - α_g , β_g , d_g	82
5.42	3 - σ Plots (Single Obstacle) - α_g , β_g , d_g	82
5.43	True versus Estimated Plots (Single Obstacle) - θ_1 , η_1 , d_1	82
5.44	3 - σ Plots (Single Obstacle) - θ_1 , η_1 , d_1	83
5.45	Single Obstacle - Range Convergence Plot	83

5.46 Single Obstacle - Commanded vs Actual Forward Velocity (m/s)	83
5.47 Single Obstacle - Commanded vs Actual Yaw Rate (rad/s)	84

List of Tables

2.1	Distance between existing LMs and new corners (in pixels)	34
5.1	Parameters used for Simulation Case I	61
5.2	Parameters used for Simulation Case II	67
5.3	Parameters used for Simulation Case III	70
5.4	Parameters used for Hardware Experiments	73

Abbreviations

RC	R emote C ontrolled
UAV	U nmanned A erial V ehicle
MEMS	M icro- E lectro- M echanical S ystems
RPF	R emotely P iloted V ehicles
UV	U nmanned V ehicle
GPS	G lobal P ositioning S ystem
LM	L andmark
EKF	E xtended K alman F ilter
FOV	F ield O f V iew
SLAM	S imultaneous L ocalization A nd M apping
UKF	U nscented K alman F ilter
RRT	R apidly E xploring R andomized T rees
MPC	M odel P redictive C ontrol
PN	P roportional N avigation
2D, 3D	2 D imensional, 3 D imensional
CW	C lockwise
CCW	C ounter C lockwise
DOF	D egree O f F reedom
VGA	V ideo G raphics A rray
ROS	R obot O perating S ystem
OBC	O n- B oard C omputer
GS	G round S tation
LOS	L ine O f S ight
PN	P roportional N avigation
LQR	L inear Q uadratic R egulator

Nomenclature

t	Experiment Run Time (s)
α_g	Azimuth Bearing Angle of the goal (rad)
β_g	Elevation Bearing Angle of the goal (rad)
d_g	Distance between UAV and goal (m)
γ_i	Azimuth Bearing Angle of the i^{th} LM (rad)
λ_i	Elevation Bearing Angle of the i^{th} LM (rad)
d_i	Distance between UAV and i^{th} LM (m)
θ_i	Difference between γ_i and α (rad)
η_i	Difference between λ_i and β (rad)
$X_g = [\alpha_g \quad \beta_g \quad d_g]^T$	UAV's orientation w.r.t the goal
$X_{l_i} = [\theta_i \quad \eta_i \quad d_{l_i}]^T$	UAV's orientation w.r.t i^{th} LM
$X = [X_g \quad X_{l1} \quad X_{l2} \quad \dots \quad X_{ln}]^T$	System states
$\dot{X}$	Motion model of the system
ζ	Zero-mean Gaussian random process with covariance Q
$y[n]$	Measurement at n^{th} time step
$\eta[n]$	Zero-mean Gaussian random process with covariance R
X_0	State vector at $t = 0$ sec
$\hat{X}$	Estimated states
T_s	Time Intervals at which internal sensor data is received (sec)
G	Motion Model Jacobian
P	System Covariance or uncertainty
h	Measurement Vector
H	Measurement Model Jacobian
L	Kalman Gain
p_g^b	3D position of goal in robot's body frame

$\mathcal{F}^b$	Body frame
$\mathcal{F}^i$	Inertial frame
$\omega_{b/i}^b$	Angular Velocity of the UAV w.r.t to $\mathcal{F}^i$ expressed in $\mathcal{F}^b$
$\begin{bmatrix} p_n & p_e & p_d \end{bmatrix}^T$	Inertial Position of the UAV in $\mathcal{F}^i$
$\begin{bmatrix} u & v & w \end{bmatrix}^T$	UAV's body frame velocity measured in $\mathcal{F}^b$
$\begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^T$	Roll, pitch and yaw angles of the UAV defined with respect to $\mathcal{F}^{v2}, \mathcal{F}^{v1}, \mathcal{F}^v$ respectively
$\begin{bmatrix} p & q & r \end{bmatrix}^T$	Roll, pitch and yaw rate measured along $\mathcal{F}^b$
$U = \begin{bmatrix} u_c & r_c \end{bmatrix}^T$	Control inputs to the UAV
d_{stop}	UAV's stopping distance from the goal (m)
α_{gdes}	Desired azimuth bearing angle with goal (rad)
N	Navigational Constant
V_r	Component of relative velocity along the LOS (m/s)
V_θ	Component of relative velocity perpendicular to LOS (m/s)
d_{obs}	Distance between UAV and obstacle
α_{obs}	Azimuth Bearing Angle of UAV w.r.t an obstacle
α_{diff}	Difference between α_{obs} and α_g
r_{col}	Collision radius of the obstacle (m)
$\mathcal{F}^{im}$	Image frame
(i_{im}, j_{im})	Image frame coordinate axis
(c_x, c_y)	Center of the image frame (pixel)
(x_g, y_g)	Position of the goal in image frame (pixel)
(x_i, y_i)	Position of the i^{th} LM in image frame (pixel)
W, H	Width, height of image frame (pixels)
dxT, dyT	Width, height of goal template (pixels)
dx_i, dy_i	Width, height of i^{th} LM template (pixels)
S_{thresh}	Threshold of template matching algorithm
Q_{corner}	Minimum accepted quality of image corners
Q_{Best}	Quality Level of most prominent corner in the scene
d_{corner}	Minimum distance in $\mathcal{F}^{im}$ for distinguishing two LMs (pixels)
F_{max}	Number of frames after which an untracked LM is discarded

Dedicated to my loving family and friends. . .

Chapter 1

Introduction

1.1 Motivation

For a majority of the 20th century, robotics has been a fascinating field of study for scientists and hobbyists alike. The first truly programmable modern robot was developed by George Devol in 1954 and installed by General Motors in 1961 to tend a die casting machine [1]. The early applications of these robots were mainly in the form of industrial manipulators and teleoperators [1, 2], later expanding into the domain of autonomous ground, aerial, underwater and space vehicles. Although experiments on pilotless flights began as early as World War I [3, 4], the boom in commercially available miniaturized aircrafts only happened in the 1960s, accelerated by the breakthroughs in computing technology [5]. These Remote Controlled (RC) planes provided everything from indoor-flyable to larger outdoor models in the form of a reasonably priced kit. More than half a century later, these small Unmanned Aerial Vehicles (UAVs) have become ubiquitous, especially due to the advent of advanced communication and Micro-Electro-Mechanical Systems (MEMS) technologies. They are being increasingly deployed for various defense and civilian applications such as reconnaissance, surveillance [6–10], pollution and weather monitoring [11–13], disaster relief [14–16], traffic monitoring [17, 18] and infrastructure inspection [19–23].

Many of these applications such as infrastructure monitoring employ Remotely Piloted Vehicles (RPVs) to achieve the task of navigation to a desired goal. The ability of a UAV to perform this mission autonomously will enable the operator to focus on more intensive tasks at hand, such as inspecting cracks and crevices from the onboard camera’s live video feed, logging data, etc. This is only possible if the UAV is able to localize itself and the goal accurately within the environment. Most Unmanned Vehicles (UVs) are dependant on Global Positioning System (GPS) data for their position, velocity and

heading information. A UAV that is equipped with a knowledge of its current GPS pose can plan paths to reach a desired goal or accomplish other complex navigational tasks. However, sole dependency on such global information can restrict applications in areas where GPS data is infrequent, unreliable or absent altogether such as indoor or underground locations, urban and natural canyons and dense woods [24]. Perhaps the best example of this is a car driving through a busy downtown where the GPS data can have high uncertainty associated with it. If the car were to solely rely on this information, even a small inaccuracy in GPS measurements will lead it to assume that it is at a different location than it actually is. Larger drifts or missed measurements might end up running this car off the road, into buildings, or worse, into pedestrians. We, as drivers, provide visual feedback that help maintain the cars on the road. Similarly, autonomous vehicles of today and the future should have the capability to navigate themselves despite loss of GPS data.

FIGURE 1.1: (a) The blue dot shows the belief regarding the device's or car's current GPS location and the circle surrounding it depicts the uncertainty associated with this belief, (b) Incorrect GPS measurement - actual location is different than the current location provided by GPS. This may be caused by Multipath Errors, Line-of-Sight (LOS) blockage, etc.

Additionally, it might not be always possible to obtain an accurate a-priori map of the environment. For example, a room in a building might be modeled according to its blueprint, however, the presence of furniture and other items makes it difficult, from a robot's perspective, to know this map beforehand. In such cases, the robot has to simultaneously map the environment and localize itself within it. A simple example of this would be a UAV flying to a goal in the same indoor environment. In the absence of GPS data, the UAV has no information regarding where it or the goal is located. This necessitates our problem formulation in a robot-centered reference frame.

1.2 Problem Description

The objective of this work is to develop an autonomous navigation algorithm for a quadrotor UAV using a robocentric reference frame. It is considered that the UAV is equipped with a monocular camera which provides it with bearing measurements from different objects in the environment. This work also implements obstacle avoidance assuming that a robust obstacle detection strategy is already in place. The UAV has to perform these tasks in a previously unknown and GPS-denied environment. This is achieved through the following steps:

- User selects a desired goal location by clicking on an object in the image from the UAV's onboard camera.
- Feature detection algorithm finds interesting features in the environment. These act as Landmarks (LMs) with respect to (w.r.t) which the UAV localizes itself.
- Goal and features are tracked using template and centroid matching techniques, respectively.
- Extended Kalman Filter (EKF) continuously estimates the UAV's states, thus, updating its belief regarding where it is located w.r.t. its surroundings.
- A path planner uses this information to help navigate the UAV to the goal, initially in an obstacle-free environment. This is followed by implementation in a course with multiple obstacles.

Some of the inherent challenges posed during the implementation of such a solution are described as follows:

1. **Goal Tracking** - Robust tracking of similar-looking objects is very difficult, hence, the desired goal should be easily visually distinguishable by the UAV's camera. When the goal is selected by the user, it might appear relatively small in the image, however, as the UAV flies towards it, the size of the goal increases. The tracking algorithm should be equipped to handle such changes. Additionally, the goal might not always be visible, for instance, it might get occluded by an obstacle or leave the Field-of-View (FOV) altogether while trying to avoid one.
2. **Landmark Detection and Tracking** - Environments consisting of limited and/or similar-looking features pose difficulty in tracking. Feature-rich environments are, therefore, a necessity for robust landmark detection and tracking. Wrongly associated landmarks can result in incorrect measurements, thereby, causing the UAV to falsely believe that it is at a different location than it actually is.

3. **EKF based Mapping** - Initializing the parameters of an EKF can be challenging as the UAV is deployed in an unknown environment. Additionally, since the EKF acts as the referential map of the environment, the size of the state vector depends on the number of LMs observed. This can increase the computational cost of our proposed algorithm.
4. **Path Planning** - This is difficult due to the fact that the environment is unknown. Controllers that require global pose information cannot be used in this case. As the UAV updates its information regarding its relative orientation with the goal, the controller should be able to use it real-time in order to accomplish the mission.

1.3 Literature Review

This Section presents a detailed review of literature that is relevant to solving this particular research problem. Each subsection introduces the reader to a specific concept within the robotics literature and then delves into the state-of-the-art within that domain. The review covers the broader aspects of a topic at first, gradually progressing to the formulations and techniques this work directly derives from. This will help illustrate the problems that this work is proposing to solve.

1.3.1 Mapping and Localization

In the field of robotics, mapping is defined as the ability of the robot to build a map or model of the environment [25]. Localization deals with determining where the robot is located w.r.t that map. In an unknown environment, the robot has neither an initial map nor an exact information about its location. In literature, this is popularly referred to as Simultaneous Localization and Mapping (SLAM). A map is needed for localizing the robot and a pose estimate is needed for mapping. This is why navigation in an unknown environment is often called a "chicken-and-egg" problem.

The problem of navigating an autonomous vehicle in an unknown environment has been widely explored in the SLAM literature [26–31]. Historically, the various SLAM problem formulations have been categorized either as Filter-based or Optimization-based approaches. The former corresponds to a two-step iterative process which is suited for online SLAM, such as Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF) [32, 33] and Particle Filter [34, 35] based techniques. Optimization-based methods comprise of techniques suited for batch-treatment of data or full SLAM, such as in GraphSLAM [36, 37] or Bundle Adjustment [38].

Following the seminal work by Smith and Cheeseman [26], EKF-SLAM was initially implemented by Mourtartier and Chatila [39] and Leonard and Durrant-Whyte [40]. It is shown in [41] that it is possible for an autonomous vehicle to start in an unknown location, and using relative measurements only (range and/or bearing), progressively build a map of the world and simultaneously compute its own pose estimates. A robust practical implementation of EKF-SLAM should be able to distinguish between true and spurious measurements, the latter of which can cause creation of fake landmarks in the environment which, in turn, can negatively affect the vehicle's localization. To this end, Hähnel et. al [42] and Bosse et. al [43] propose data association techniques that maximize the perceptual distinctiveness of landmarks. Any Filter-based SLAM approaches add observed landmarks into the state vector, i.e., the size of the map increases progressively with the number of landmarks in the environment. As such, latter research in this area has focused on achieving scalability by decomposing the global map into a series of submaps [44–46]. Williams et. al [47] provides results for underwater mapping using multiple local submaps that are intermittently fused into the global map of the environment.

In recent years, vision-based SLAM research has been gaining popularity [35, 48], primarily, due to the high information to weight ratio of camera sensors that can help in obstacle detection, place recognition and visual servoing. The inconsistency and unobservability of the world-centric SLAM solution is studied in [49, 50]. It is shown in [51] that formulating the estimation problem in Modified Polar (MP) coordinates leads to an EKF which is both stable and asymptotically unbiased. The concept of robocentric mapping is introduced [52] and shown to better deal with linearization errors than the traditional world-centric formulation. It is shown that linearizing the measurement model significantly improves the consistency of EKF-SLAM. Furthermore, [53] defines an approach of bearing-only SLAM using the modified spherical coordinate system and [54] uses a robot-centered reference frame for accomplishing the task of autonomous navigation of a ground robot in a GPS-denied environment. It is shown in [54] that goal-aware robocentric mapping for a ground vehicle causes its state estimates to converge significantly faster than the robocentric method proposed in [53]. However, [54] uses a nonlinear measurement model which can, in turn, affect the consistency of EKF [53]. This research work extends the goal-aware robocentric formulation developed in [54] for a quadrotor UAV and uses a direct bearing measurement as updates to the EKF. This generates a linear measurement model and reduces complexity in a system which already has a highly nonlinear motion model.

1.3.2 Path Planning

Path or motion planning is defined as the process of formulating a robot's navigation strategy such that it can satisfy its own motion constraints and optimize certain mission criteria such as path length or time. Other mission criteria might include the ability to maintain a certain distance or bearing with respect to the goal as in the case of this research work (see Section 2.3). Path planning is a prerequisite to exploration which has been a primary application domain in robotic systems development. It has seen extensive applications related to planetary/lunar exploration [55, 56], search and rescue [57, 58], abandoned mine mapping [59, 60], etc.

In the context of unknown environments, path planning has to take into account the unavailability of any global positioning data in addition to any process and sensor uncertainties. Batalin and Sukhatme [61] explain a technique for exploration wherein a robot drops different sensors to aid the localization process. Huang et. al [62] and Bry et. al [63] use Rapidly exploring Randomized Trees (RRT) to solve motion planning under uncertainties. Sadat et. al [64] proposes a technique for path planning which maintains visual contacts with feature-rich areas. Leung et. al [65] addresses optimal trajectory planning in SLAM using Model Predictive Control (MPC). An optimization problem for path planning is solved in [54] by associating higher costs to paths containing obstacles and lower costs to paths that minimize the uncertainty of the goal's position while the robot drives towards it.

Proportional Navigation (PN) guidance has been widely used in missile guidance literature for the purpose of intercepting targets [66]. It is easy to mechanize and can take corrective action from the very beginning. It has been successfully implemented for robust trajectory-tracking [67] and collision avoidance of UAVs [68, 69]. These techniques solve collision avoidance with a target under different initial conditions. The objective of this research is to achieve static obstacle avoidance based on a reactive strategy developed in [70] to help prevent collisions as the UAV flies to its goal.

1.3.3 Visual Tracking

Visual tracking is defined as the ability of a robot to follow changes in the position of an object within its camera's Field-of-View (FOV). Such changes may occur either due to its own motion or due to the motion of the object. Visual tracking is accomplished by first detecting an object and then, establishing correspondence between the object's instances across consecutive image frames. Historically, these techniques can be classified into point, kernel and silhouette based approaches [71]. Point detectors such as

Harris corner detector [72], Shi Tomasi [73] and SIFT detector [74] are some popular techniques which are used to extract the centroids of objects that may have an expressive texture with respect to its locality. They are, then, tracked using deterministic or stochastic methods of object correspondence based on a combination of various motion constraints such as proximity, maximum velocity, common motion and rigidity [71, 75]. Most real-time object detectors are, however, computationally intensive, thus, leaving little time for further processing of the objects or the image in general. Significant research has, therefore, focused on gaining increased speeds of computation using learning, classification [76, 77] and dimensionality reduction techniques [78–80].

Among all object features such as color, edge, texture and optical flow, color is the most widely used feature for tracking [81–84]. It is particularly robust when the environment is affected by no or limited illumination changes and similar-colored objects do not impair the efficiency of the tracker. Another common form of object tracking is via the use of density-based appearance models such as mean-shift [85] and template tracking algorithms [86–88]. These are widely used because of their relative simplicity and low computational cost. Detailed survey including several other object detection and tracking strategies are presented in [71, 89].

1.4 Summary of Contributions

Based on the objective proposed in Section 1.2, this Section covers the contributions made by this research work. They can be summarized as follows:

- Based on [54], a goal-aware robocentric mapping algorithm is developed for a multirotor UAV. The center of the UAV is considered to be the origin of the map, hence, all the goal and landmarks are represented with respect to the UAV's body frame. The orientation of the landmarks are modeled with reference to the goal as described in Section 2.2.1. An EKF comprising of a bearing-only measurement model is used to estimate the relative orientation of the UAV w.r.t to the goal and landmarks.
- A detailed 2D observability analysis of the system is performed and shown to match with simulation results for three different cases. This work establishes the advantage of 'goal-aware' formulation by comparing the trace of the observability Gramian obtained for Case 3.4 using both the formulation proposed in this Thesis and in [53].
- A combined PN-guidance and obstacle avoidance controller is developed and implemented to meet the UAV's navigation objective whilst avoiding obstacle(s) along

its path. Furthermore, a velocity controller is developed in simulation to map the high-level controller outputs to commands that can be interpreted by the UAV's autopilot. This is done so that the simulation UAV emulates the actual UAV used for hardware experiments.

- A visual tracker is developed to track the positions of the goal and landmarks in the image frame. These pixel locations are used to calculate their respective bearing measurements. A multi-scale template matching algorithm is used to accommodate increasing size of goal as the UAV maneuvers towards it. This algorithm is also made color and illumination-invariant such that goal objects of variable intensity can be tracked. A simple heuristic-based data association algorithm ensures that existing landmarks are not wrongly associated with each other or spurious measurements from the environment.
- The proposed algorithm is validated through extensive simulation and hardware results. We present results for direct navigation to goal without any obstacles. This is followed by cases which establish the ability of this algorithm to robustly track and accurately estimate the goal's states even after it has been forced out of the camera's FOV due to the presence of obstacles. These algorithms are implemented and tested using AR.Drone 2.0 platform.

1.5 Thesis Layout

From this point on, the Thesis is organized as follows: Chapter 2 delves into the Problem Methodology and provides a detailed explanation of State Modeling, Estimation, Path Planning and Visual Tracking relevant to this problem solution. Chapter 3 analyzes the conditions required for the system to be observable, followed by which the system setup is described in Chapter 4. It highlights the various hardware and software equipment used in validating our proposed solution. The simulation and experimental results are presented and discussed in Chapter 5. Finally, the work is summarized and possible future scope of research is presented in Chapter 6.

Chapter 2

Problem Methodology

2.1 State Definitions

In the absence of any global positioning data, it is very difficult to ascertain the position of a UAV, its goal and other objects of interest, called Landmarks (LMs) with respect to a fixed coordinate frame. Therefore, we formulate the problem of navigation in an unknown environment in robocentric frame, where the origin is located at the center of mass of the UAV. All other objects in the environment are defined based on their relative orientation with respect to the UAV. Figure. 2.1 represents this formulation with one goal and one landmark.

FIGURE 2.1: State Modeling

Here, α_g and β_g are the azimuth and elevation angles (in radians) of the goal respectively and d_g is the distance of the goal from the UAV (in meters). Similarly, d_i is the distance between the UAV and the i^{th} Landmark. θ_i is the angle between the projection of the goal point on i^b-j^b plane (blue dashed line) and the projection of the i^{th} LM point on the i^b-j^b plane. η_i is the difference between the elevation angle of LM_i and that of goal.

$$\gamma_i = \alpha_g + \theta_i, \quad \lambda_i = \beta_g + \eta_i \quad (2.1)$$

This simply means that what α_g and β_g are to the goal, γ_i and λ_i are to the i^{th} LM respectively. However, we consider θ_i and η_i to be the i^{th} LM's states, thus, maintaining the goal as the anchor of the map. Since this represents the evolution of the LMs with respect to the goal, we term this formulation as 'goal-aware' robocentric mapping. At any point of time, the map of the environment is defined by the state of these LMs and the goal relative to the UAV. The state vector can be represented as:

$$\begin{aligned} X &= [\alpha_g \ \beta_g \ d_g \ \theta_1 \ \eta_1 \ d_1 \ \dots \ \theta_n \ \eta_n \ d_n]^T \\ &= [X_g \ X_{l_1} \ X_{l_2} \ \dots \ X_{l_n}]^T \end{aligned} \quad (2.2)$$

where,

$$X_g = \begin{bmatrix} \alpha_g \\ \beta_g \\ d_g \end{bmatrix}, X_{l_1} = \begin{bmatrix} \theta_1 \\ \eta_1 \\ d_{l_1} \end{bmatrix}, \dots \quad X_{l_n} = \begin{bmatrix} \theta_n \\ \eta_n \\ d_{l_n} \end{bmatrix} \quad (2.3)$$

Here, X_g and X_{l_i} represent the goal and i^{th} landmark states respectively. Thus, the system has a total of $3n + 3$ states, where n is the number of landmarks.

2.2 State Estimation

In order to navigate successfully to its goal, the UAV needs to localize itself with respect to the goal accurately, i.e., it should have very little uncertainty regarding its knowledge of α_g , β_g and d_g . While α_g and β_g are measured directly using the UAV's bearing sensor, there is no direct measurement available regarding the distance to the goal. The UAV's camera processes the images at a much slower rate as compared to the speed it is traveling at. Measurements after certain intervals of time might not prove sufficient for the UAV's navigation. Additionally, the goal might occasionally leave the camera's Field-of-View (FOV) due to changes in its trajectory, or even if it is within the FOV,

the view might be occluded due to the presence of LMs or other obstacles along the path. Thus, estimating the UAV's orientation w.r.t. the goal becomes an absolute necessity throughout the duration of the mission. Since accurate localization is difficult to achieve using only one set of measurement, the relative orientation of the UAV with respect to certain LMs are also estimated. We use an Extended Kalman Filter (EKF) for this purpose. An EKF predicts the states given in Eq. (2.2) based on the goal and LM motion models (Eqs. (2.17) and (2.19) respectively) and updates its belief as and when new measurements are received. For a nonlinear system such as ours, the system dynamics are given by

$$\dot{X} = f(X, U) + \xi \quad (2.4)$$

$$y[n] = h(X[n], U[n]) + \eta[n] \quad (2.5)$$

where, $y[n] = y(t_n)$ is the measurement at n^{th} time step, $X[n] = X(t_n)$ is the n^{th} sample of X , $\eta[n]$ is the measurement noise at time t_n , ξ is a zero-mean Gaussian random process with covariance Q , and $\eta[n]$ is zero-mean Gaussian random variable with covariance R . The random noise ξ is called process noise and models the disturbances that can act on the system. On the other hand, η represents the noise in the sensor measurements.

The EKF algorithm is outlined as follows:

Algorithm 1 Continuous Extended Kalman Filter

- 1: Initialize $\hat{X} = X_0$
 - 2: For each sample time T_s :
 - 3: **for** $i = 1$ to N **do** Prediction Step
 - 4: $\hat{X} = \hat{X} + (\frac{T_s}{N}) f(\hat{X}, U)$
 - 5: $G = \frac{\partial f(\hat{X}, U)}{\partial X}$ (Jacobian of Eqs. (2.17) and (2.19) with respect to Eq. (2.2))
 - 6: $P = P + (\frac{T_s}{N}) (GP + PG^T + Q)$
 - 7: **end for**
 - 8: **If** bearing measurement has been received from sensor i **then** bearing update
 - 9: $H_i = \frac{\partial h(\hat{X}, U)}{\partial X}$ (jacobian of Eq. (2.21))
 - 10: $L_i = PH_i^T (R_i + H_iPH_i^T)^{-1}$
 - 11: $P = (I - L_iH_i)P$
 - 12: $\hat{X} = \hat{X} + L_i(y_i - h(\hat{X}, U))$
 - 13: **end if**
-

where, $f(\hat{X}, U)$ is the state motion model which shows how the states evolve with time, G and H are the motion model and measurement model Jacobians respectively. Finally,

L_i is the Kalman Gain and I refers to an identity matrix. The following subsections provide detailed derivations for terms used in Algorithm 1.

2.2.1 Motion Model

Based Figure 2.1, the position of the goal in the UAV's body frame $\mathcal{F}^b$ is:

$$\mathbf{p}_g^b = \begin{bmatrix} d_g \cos \alpha_g \cos \beta_g \\ d_g \cos \alpha_g \sin \beta_g \\ d_g \sin \beta_g \end{bmatrix} \quad (2.6)$$

Here, $\mathbf{p}_g^b$ is the vector connecting the UAV to the goal. Using this, the vector from the goal to the UAV can be represented as:

$$\mathbf{p}_r^b = \begin{bmatrix} -d_g \cos \alpha_g \cos \beta_g \\ -d_g \cos \alpha_g \sin \beta_g \\ -d_g \sin \beta_g \end{bmatrix} \quad (2.7)$$

In order to calculate the time derivatives of the goal states, we need to calculate the derivative of the vector $\mathbf{p}$. Using [90], the time derivative of $\mathbf{p}_r$ can be written as:

$$\frac{d}{dt_i} \mathbf{p}_r^b = \frac{d}{dt_b} \mathbf{p}_r^b + \omega_{b/i} \times \mathbf{p}_r^b \quad (2.8)$$

Here, $\frac{d}{dt_i} \mathbf{p}_r^b$ and $\frac{d}{dt_b} \mathbf{p}_r^b$ are the first-order time derivative of $\mathbf{p}_r$ with respect to the inertial frame, $\mathcal{F}^i$ and body frame, $\mathcal{F}^b$ respectively. Eq. (2.8) can be rewritten as:

$$\frac{d}{dt_b} \mathbf{p}_r^b = \frac{d}{dt_i} \mathbf{p}_r^b - \omega_{b/i}^b \times \mathbf{p}_r^b \quad (2.9)$$

Differentiating Eq. (2.7) with respect to time,

$$\begin{aligned} \frac{d}{dt_b} \mathbf{p}_r^b &= \begin{bmatrix} d_g \dot{\alpha}_g \sin \alpha_g \cos \beta_g + d_g \dot{\beta}_g \cos \alpha_g \sin \beta_g - \dot{d}_g \cos \alpha_g \cos \beta_g \\ -d_g \dot{\alpha}_g \cos \alpha_g \cos \beta_g + d_g \dot{\beta}_g \sin \alpha_g \sin \beta_g - \dot{d}_g \sin \alpha_g \cos \beta_g \\ -d_g \dot{\beta}_g \cos \beta_g - \dot{d}_g \sin \beta_g \end{bmatrix} \\ &= M \begin{bmatrix} \dot{\alpha}_g \\ \dot{\beta}_g \\ \dot{d}_g \end{bmatrix} \end{aligned} \quad (2.10)$$

where,

$$M = \begin{bmatrix} d_g \sin \alpha_g \cos \beta_g & d_g \cos \alpha_g \sin \beta_g & -\cos \alpha_g \cos \beta_g \\ -d_g \cos \alpha_g \cos \beta_g & d_g \sin \alpha_g \sin \beta_g & -\sin \alpha_g \sin \beta_g \\ 0 & -d_g \cos \beta_g & -\sin \beta_g \end{bmatrix} \quad (2.11)$$

The first term on the right-hand side of Eq. (2.9) is the linear velocity of the UAV along its $\mathbf{i}^b$, $\mathbf{j}^b$ and $\mathbf{k}^b$ axis respectively, as represented in $\mathcal{F}^i$. While a UAV can move sideways as well as in the vertical direction, we only consider the forward velocity in order to reduce the complexity of the motion model. Therefore, we can write

$$\frac{d}{dt_i} \mathbf{p}_r^b = \begin{bmatrix} u \\ 0 \\ 0 \end{bmatrix} \quad (2.12)$$

where, u is the linear velocity along $\mathbf{i}^b$ axis, defined in m/s . $\omega_{b/i}^b$ is the angular velocity of the UAV with respect to $\mathcal{F}^i$ expressed in $\mathcal{F}^b$.

$$\omega_{b/i}^b = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \quad (2.13)$$

Substituting Eqs. (2.7), (2.10), (2.12) and (2.13) in Eq. (2.9), we can write the following:

$$\begin{aligned} M \begin{bmatrix} \dot{\alpha}_g \\ \dot{\beta}_g \\ \dot{d}_g \end{bmatrix} &= \begin{bmatrix} u \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \begin{bmatrix} -d_g \cos \alpha_g \cos \beta_g \\ -d_g \cos \alpha_g \sin \beta_g \\ -d_g \sin \beta_g \end{bmatrix} \\ &= \begin{bmatrix} u + d_g \cos \alpha_g \cos \beta_g \\ d_g \cos \alpha_g \sin \beta_g \\ d_g \sin \beta_g \end{bmatrix} \end{aligned} \quad (2.14)$$

Taking M to the right-hand side,

$$\begin{bmatrix} \dot{\alpha}_g \\ \dot{\beta}_g \\ \dot{d}_g \end{bmatrix} = M^{-1} \begin{bmatrix} u + d_g \cos \alpha_g \cos \beta_g \\ d_g \cos \alpha_g \sin \beta_g \\ d_g \sin \beta_g \end{bmatrix} \quad (2.15)$$

where,

$$M^{-1} = \begin{bmatrix} \frac{\sin \alpha_g}{d_g \cos \beta_g} & \frac{\cos \alpha_g}{d_g \cos \beta_g} & 0 \\ \frac{\cos \alpha_g \sin \beta_g}{d_g} & \frac{\sin \alpha_g \sin \beta_g}{d_g} & -\frac{\cos \beta_g}{d_g} \\ -\cos \alpha_g \cos \beta_g & -\sin \alpha_g \cos \beta_g & -\sin \beta_g \end{bmatrix} \quad (2.16)$$

Substituting Eq. (2.16) in Eq. (2.15) and multiplying them, we get the goal motion model as:

$$f(X_g, U) = \begin{bmatrix} \dot{\alpha}_g \\ \dot{\beta}_g \\ \dot{d}_g \end{bmatrix} = \begin{bmatrix} \frac{u \sin \alpha_g}{d_g \cos \beta_g} + p \sin \alpha_g \cos \alpha_g - q \sin \alpha_g \cos \alpha_g \\ \frac{u \cos \alpha_g \sin \beta_g}{d_g} - p \sin \beta_g \cos \beta_g \cos^2 \alpha_g + q \sin \beta_g \cos \beta_g \sin^2 \alpha_g - \\ r \sin \beta_g \cos \beta_g \\ -u \cos \alpha_g \cos \beta_g - d_g p \cos^2 \alpha_g \cos^2 \beta_g - d_g q \cos^2 \beta_g \sin^2 \alpha_g - \\ d_g r \sin^2 \beta_g \end{bmatrix} \quad (2.17)$$

Taking the time derivative of Eq. (2.1) and bringing the landmark states to the left-hand side, we get

$$\dot{\theta}_i = \dot{\gamma}_i - \dot{\alpha}_g, \quad \dot{\eta}_i = \dot{\lambda}_i - \dot{\beta}_g \quad (2.18)$$

The equations for $\dot{\gamma}_i$ and $\dot{\lambda}_i$ will be the same as that of $\dot{\alpha}_g$ and $\dot{\beta}_g$ in Eq. (2.17), except of course, the angles α_g and β_g will be replaced by the angles γ_i and λ_i . Therefore, the landmark motion model can be defined as:

$$f(X_{l_i}, U) = \begin{bmatrix} \dot{\theta}_i \\ \dot{\eta}_i \\ \dot{d}_i \end{bmatrix} = \begin{bmatrix} \frac{u \sin(\alpha_g + \theta_i)}{d_i \cos(\beta_g + \eta_i)} + p \sin(\alpha_g + \theta_i) \cos(\alpha_g + \theta_i) \\ -q \sin(\alpha_g + \theta_i) \cos(\alpha_g + \theta_i) - \dot{\alpha}_g \\ \frac{u \cos(\alpha_g + \theta_i) \sin(\beta_g + \eta_i)}{d_i} + p \cos^2(\alpha_g + \theta_i) \sin(\beta_g + \eta_i) \cos(\beta_g + \eta_i) \\ + q \sin^2(\alpha_g + \theta_i) \sin(\beta_g + \eta_i) \cos(\beta_g + \eta_i) \\ -r \sin(\beta_g + \eta_i) \cos(\beta_g + \eta_i) - \dot{\beta}_g \\ -u \cos(\alpha_g + \theta_i) \cos(\beta_g + \eta_i) - d_i p \cos^2(\alpha_g + \theta_i) \cos^2(\beta_g + \eta_i) \\ -d_i q \sin(\alpha_g + \theta_i) \cos^2(\beta_g + \eta_i) - d_i r \sin^2(\beta_g + \eta_i) \end{bmatrix} \quad (2.19)$$

2.2.2 Motion Model Jacobian

The motion model Jacobian G can be written as:

$$G = \begin{bmatrix} G_g & 0 & 0 & \dots & 0 \\ G_{l_1,g} & G_{l_1,l_1} & 0 & \dots & 0 \\ G_{l_2,g} & 0 & G_{l_2,l_2} & \dots & 0 \\ \vdots & \vdots & 0 & \ddots & 0 \\ G_{l_n,g} & 0 & \dots & 0 & G_{l_n,l_n} \end{bmatrix} \quad (2.20)$$

where,

$$G_g = \begin{bmatrix} \frac{\partial g_{\alpha g}}{\partial \alpha_g} & \frac{\partial g_{\alpha g}}{\partial \beta_g} & \frac{\partial g_{\alpha g}}{\partial d_g} \\ \frac{\partial g_{\beta g}}{\partial \alpha_g} & \frac{\partial g_{\beta g}}{\partial \beta_g} & \frac{\partial g_{\beta g}}{\partial d_g} \\ \frac{\partial g_{d_g}}{\partial \alpha_g} & \frac{\partial g_{d_g}}{\partial \beta_g} & \frac{\partial g_{d_g}}{\partial d_g} \end{bmatrix}, \quad G_{l_i,g} = \begin{bmatrix} \frac{\partial g_{\theta i}}{\partial \alpha_g} & \frac{\partial g_{\theta i}}{\partial \beta_g} & \frac{\partial g_{\theta i}}{\partial d_g} \\ \frac{\partial g_{\eta i}}{\partial \alpha_g} & \frac{\partial g_{\eta i}}{\partial \beta_g} & \frac{\partial g_{\eta i}}{\partial d_g} \\ \frac{\partial g_{d_i}}{\partial \alpha_g} & \frac{\partial g_{d_i}}{\partial \beta_g} & \frac{\partial g_{d_i}}{\partial d_g} \end{bmatrix}, \quad G_{l_i,l_i} = \begin{bmatrix} \frac{\partial g_{\theta i}}{\partial \theta_i} & \frac{\partial g_{\theta i}}{\partial \eta_i} & \frac{\partial g_{\theta i}}{\partial d_i} \\ \frac{\partial g_{\eta i}}{\partial \theta_i} & \frac{\partial g_{\eta i}}{\partial \eta_i} & \frac{\partial g_{\eta i}}{\partial d_i} \\ \frac{\partial g_{d_i}}{\partial \theta_i} & \frac{\partial g_{d_i}}{\partial \eta_i} & \frac{\partial g_{d_i}}{\partial d_i} \end{bmatrix}$$

The goal-state Jacobian, G_g , can be written as:

$$\frac{\partial g_{\alpha g}}{\partial \alpha_g} = \frac{u \cos \alpha_g}{d_g \cos \beta_g} - \sin^2 \alpha_g (p - q) + \cos^2 \alpha_g (p - q)$$

$$\frac{\partial g_{\alpha g}}{\partial \beta_g} = \frac{u \sin \alpha_g \sin \beta_g}{d_g \cos^2 \beta_g}$$

$$\frac{\partial g_{\alpha g}}{\partial d_g} = \frac{-u \sin \alpha_g}{d_g^2 \cos \beta_g}$$

$$\frac{\partial g_{\beta g}}{\partial \alpha_g} = \frac{-u \sin \alpha_g \sin \beta_g}{d_g} - 2 \sin \alpha_g \cos \alpha_g \sin \beta_g \cos \beta_g (p - q)$$

$$\frac{\partial g_{\beta g}}{\partial \beta_g} = \frac{u \cos \alpha_g \cos \beta_g}{d_g} - (\sin^2 \beta_g - \cos^2 \beta_g)(p \cos^2 \alpha_g + q \sin^2 \alpha_g - r)$$

$$\frac{\partial g_{\beta g}}{\partial d_g} = \frac{-u \cos \alpha_g \sin \beta_g}{d_g^2}$$

$$\frac{\partial g_{d_g}}{\partial \alpha_g} = u \sin \alpha_g \cos \beta_g + 2d_g \sin \alpha_g \cos \alpha_g \cos^2 \beta_g (p - q)$$

$$\frac{\partial g_{d_g}}{\partial \beta_g} = u \cos \alpha_g \sin \beta_g + 2d_g \sin \beta_g \cos \beta_g (p \cos^2 \alpha_g + q \sin^2 \alpha_g - r)$$

$$\frac{\partial g_{d_g}}{\partial d_g} = -p \cos^2 \alpha_g \cos^2 \beta_g - q \sin^2 \alpha_g \cos^2 \beta_g - r \sin^2 \beta_g$$

The landmark state Jacobian, $G_{l_i, g}$, can be written as:

$$\frac{\partial g_{\theta_i}}{\partial \alpha_g} = \frac{u \cos(\alpha_g + \theta_i)}{d_i \cos(\beta_g + \eta_i)} - \sin^2(\alpha_g + \theta_i)(p - q) + \cos^2(\alpha_g + \theta_i)(p - q) - \frac{\partial g_{\alpha_g}}{\partial \alpha_g}$$

$$\frac{\partial g_{\theta_i}}{\partial \beta_g} = \frac{u \sin(\alpha_g + \theta_i) \sin(\beta_g + \eta_i)}{d_i \cos^2(\beta_g + \eta_i)} - \frac{\partial g_{\alpha_g}}{\partial \beta_g}$$

$$\frac{\partial g_{\theta_i}}{\partial d_g} = \frac{u \sin \alpha_g}{d_g^2 \cos \beta_g}$$

$$\begin{aligned} \frac{\partial g_{\eta_i}}{\partial \alpha_g} &= \frac{-u \sin(\alpha_g + \theta_i) \sin(\beta_g + \eta_i)}{d_i} \\ &\quad - 2 \sin(\alpha_g + \theta_i) \cos(\alpha_g + \theta_i) \sin(\beta_g + \eta_i) \cos(\beta_g + \eta_i)(p - q) - \frac{\partial g_{\beta_g}}{\partial \alpha_g} \end{aligned}$$

$$\begin{aligned} \frac{\partial g_{\eta_i}}{\partial \beta_g} &= \frac{u \cos(\alpha_g + \theta_i) \cos(\beta_g + \eta_i)}{d_i} \\ &\quad - (\sin^2(\beta_g + \eta_i) - \cos^2(\beta_g + \eta_i))(p \cos^2(\alpha_g + \theta_i) + q \sin^2(\alpha_g + \theta_i) - r) - \frac{\partial g_{\beta_g}}{\partial \beta_g} \end{aligned}$$

$$\frac{\partial g_{\eta_i}}{\partial d_g} = \frac{u \cos \alpha_g \sin \beta_g}{d_g^2}$$

$$\frac{\partial g_{d_i}}{\partial \alpha_g} = u \sin(\alpha_g + \theta_i) \cos(\beta_g + \eta_i) + 2d_i \sin(\alpha_g + \theta_i) \cos(\alpha_g + \theta_i) \cos^2(\beta_g + \eta_i)(p - q) - \frac{\partial g_{d_g}}{\partial \alpha_g}$$

$$\begin{aligned} \frac{\partial g_{d_i}}{\partial \beta_g} &= u \cos(\alpha_g + \theta_i) \sin(\beta_g + \eta_i) \\ &\quad + 2d_i \sin(\beta_g + \eta_i) \cos(\beta_g + \eta_i)(p \cos^2(\alpha_g + \theta_i) + q \sin^2(\alpha_g + \theta_i) - r) - \frac{\partial g_{d_g}}{\partial \beta_g} \end{aligned}$$

$$\frac{\partial g_{d_i}}{\partial d_g} = 0$$

The landmark state Jacobian, G_{l_i, l_i} , can be written as:

$$\frac{\partial g_{\theta_i}}{\partial \theta_i} = \frac{u \cos(\alpha_g + \theta_i)}{d_i \cos(\beta_g + \eta_i)} - \sin^2(\alpha_g + \theta_i)(p - q) + \cos^2(\alpha_g + \theta_i)(p - q)$$

$$\frac{\partial g_{\theta_i}}{\partial \eta_i} = \frac{u \sin(\alpha_g + \theta_i) \sin(\beta_g + \eta_i)}{d_i \cos^2(\beta_g + \eta_i)}$$

$$\frac{\partial g_{\theta_i}}{\partial d_i} = \frac{-u \sin(\alpha_g + \theta_i)}{d_i^2 \cos(\beta_g + \eta_i)}$$

$$\frac{\partial g_{\eta_i}}{\partial \theta_i} = \frac{-u \sin(\alpha_g + \theta_i) \sin(\beta_g + \eta_i)}{d_i} - 2 \sin(\alpha_g + \theta_i) \cos(\alpha_g + \theta_i) \sin(\beta_g + \eta_i) \cos(\beta_g + \eta_i)(p - q)$$

$$\frac{\partial g_{\eta_i}}{\partial \eta_i} = \frac{u \cos(\alpha_g + \theta_i) \cos(\beta_g + \eta_i)}{d_i} - (\sin^2(\beta_g + \eta_i) - \cos^2(\beta_g + \eta_i))(p \cos^2(\alpha_g + \theta_i) + q \sin^2(\alpha_g + \theta_i) - r)$$

$$\frac{\partial g_{\eta_i}}{\partial d_i} = \frac{-u \cos(\alpha_g + \theta_i) \sin(\beta_g + \eta_i)}{d_i^2}$$

$$\frac{\partial g_{d_i}}{\partial \theta_i} = u \sin(\alpha_g + \theta_i) \cos(\beta_g + \eta_i) + 2d_i \sin(\alpha_g + \theta_i) \cos(\alpha_g + \theta_i) \cos^2(\beta_g + \eta_i)(p - q)$$

$$\frac{\partial g_{d_i}}{\partial \eta_i} = u \cos(\alpha_g + \theta_i) \sin(\beta_g + \eta_i) + 2d_i \sin(\beta_g + \eta_i) \cos(\beta_g + \eta_i)(p \cos^2(\alpha_g + \theta_i) + q \sin^2(\alpha_g + \theta_i) - r)$$

$$\frac{\partial g_{d_i}}{\partial d_i} = -p \cos^2(\alpha_g + \theta_i) \cos^2(\beta_g + \eta_i) - q \cos^2(\beta_g + \eta_i) \sin^2(\alpha_g + \theta_i) - r \sin^2(\beta_g + \eta_i)$$

2.2.3 Measurement Model and Measurement Jacobian

A monocular camera is used to track the goal and the LMs in the environment. The camera provides two measurements each for the goal and individual LMs. These are the azimuth and elevation bearing angles. The measurement model can be defined as:

$$h = [\alpha_g \ \beta_g \ \gamma_1 \ \lambda_1 \ \dots \ \gamma_n \ \lambda_n]^T \quad (2.21)$$

It should be noted that α_g, β_g in Eq. (2.21) may not necessarily have the same values as α_g, β_g in Eq. (2.2). These notations are used to signify that these variables denote the same physical entities (i.e., azimuth and elevation bearing angle of the UAV w.r.t goal, respectively), irrespective of whether they are being estimated or measured. The size of the measurement vector is $2n + 2$, where n is the number of landmarks. The bearing angles are calculated based on the object's pixel location in the image frame and can be given by [91] :

$$\alpha_g = \tan^{-1} \left(\frac{c_x - x_g}{f_x} \right), \quad \beta_g = \tan^{-1} \left(\frac{c_y - y_g}{f_y} \right) \quad (2.22)$$

$$\gamma_i = \tan^{-1} \left(\frac{c_x - x_i}{f_x} \right), \quad \lambda_i = \tan^{-1} \left(\frac{c_y - y_i}{f_y} \right) \quad (2.23)$$

where, f_x, f_y are the focal length in x (horizontal) and y (vertical) directions respectively. (c_x, c_y) and (x_g, y_g) are (x, y) coordinates of the center and the goal in the image frame, respectively. Finally, x_i and y_i are the locations of the i^{th} landmark in the image plane. The measurement Jacobian is used to update the states as well the covariance associated with them. We know that the measurement model Jacobian H is the linearization of the measurement vector at the current states of the system, and is defined as:

$$H = \frac{\partial h}{\partial X} \quad (2.24)$$

Similar to the motion model Jacobian in Eq. (2.20), H can be defined as:

$$H = \begin{bmatrix} H_g & 0 & 0 & \dots & 0 \\ H_{l_1,g} & H_{l_1,l_1} & 0 & \dots & 0 \\ H_{l_2,g} & 0 & H_{l_2,l_2} & \dots & 0 \\ \vdots & \vdots & 0 & \ddots & 0 \\ H_{l_n,g} & 0 & \dots & 0 & H_{l_n,l_n} \end{bmatrix} \quad (2.25)$$

where,

$$H_g = \begin{bmatrix} \frac{\partial h_{\alpha g}}{\partial \alpha_g} & \frac{\partial h_{\alpha g}}{\partial \beta_g} & \frac{\partial h_{\alpha g}}{\partial d_g} \\ \frac{\partial h_{\beta g}}{\partial \alpha_g} & \frac{\partial h_{\beta g}}{\partial \beta_g} & \frac{\partial h_{\beta g}}{\partial d_g} \end{bmatrix}, H_{l_i,g} = \begin{bmatrix} \frac{\partial h_{\gamma i}}{\partial \alpha_g} & \frac{\partial h_{\gamma i}}{\partial \beta_g} & \frac{\partial h_{\gamma i}}{\partial d_g} \\ \frac{\partial h_{\lambda i}}{\partial \alpha_g} & \frac{\partial h_{\lambda i}}{\partial \beta_g} & \frac{\partial h_{\lambda i}}{\partial d_g} \end{bmatrix}, H_{l_i,l_i} = \begin{bmatrix} \frac{\partial h_{\gamma i}}{\partial \theta_i} & \frac{\partial h_{\gamma i}}{\partial \eta_i} & \frac{\partial h_{\gamma i}}{\partial d_i} \\ \frac{\partial h_{\lambda i}}{\partial \theta_i} & \frac{\partial h_{\lambda i}}{\partial \eta_i} & \frac{\partial h_{\lambda i}}{\partial d_i} \end{bmatrix}$$

$$\Rightarrow H_g = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad H_{l_i,g} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad H_{l_i,l_i} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

H is used in Line 10 of Algorithm 1 to calculate the Kalman gain L_i , which is basically a measure of how much the states of the system should be changed based on a measurement input. L_i is then used to update P and $\hat{X}$, as can be seen from Lines 11 and 12 of Algorithm 1.

2.3 Path Planning

2.3.1 Navigation to Goal

Successfully solving the estimation problem allows us to develop a controller which uses the estimated states of the UAV to help navigate it towards a user-defined goal. Since the proposed solution is developed in order to aid applications such as infrastructure inspection, the objective is not for the UAV to hit the goal, rather, it should reach within a certain predefined stopping distance (d_{stop}) and bearing (α_{gdes}) of the goal such that the operators can analyze the structural deficiencies up-close. For the purposes of our controller, we define our problem in 2-D plane, i.e., we control the UAV's motion similar to any ground vehicle by manipulating its forward velocity and turn rate. Thus, the control inputs to the system can be written as:

$$U = \begin{bmatrix} u_c & r_c \end{bmatrix}^T \quad (2.26)$$

where, u_c is the commanded forward velocity (m/s) and r_c is the commanded yaw rate (rad/s). This is directly related to our problem definition which specifies that the user selects the goal dynamically at the beginning of the mission, which means that the maximum altitude difference between the goal and the UAV is limited by the Field-of-View (FOV) of its camera. Therefore, an altitude change is not commanded to the UAV and as long as it can navigate to the specified d_{stop} and α_{gdes} , the mission objective is considered to have been met. The control objectives in an obstacle-free environment can be defined as:

- Change the orientation of the UAV such that it maintains a desired bearing angle to the goal. This can be achieved by changing the UAV's heading, i.e., commanding its r_c .

- Move forward to the goal until a desired stopping distance or a desired elevation angle with the goal (β_g) is reached. It is necessary to impose the latter condition because as $d_g \downarrow$, $\text{mod}(\beta_g) \uparrow$. In order to successfully monitor or inspect infrastructures, the requirement to maintain $\beta_g < \text{Vertical FOV angle}$ is obvious. This can be obtained by commanding a desired forward velocity, u_c to the UAV.
- Once the desired α_g and d_g (or β_g) are obtained, apply no control commands to the UAV, i.e., $u_c = r_c = 0$.

As mentioned in Chapter 1, we use Proportional Navigation (PN) guidance law to achieve the desired control objective. PN guidance law is commonly used in some form or another by most homing target missiles [66]. The guidance is in the form of a lateral acceleration (commonly called latax) command which ensures that the rate of rotation of the vehicle velocity vector is proportional to the rate of rotation of the LOS (Line-of-Sight), i.e.,

$$\dot{\psi} = N\dot{\theta} \quad (2.27)$$

where, N is called the Navigational constant (generally between 1 and 5), ψ is the vehicle heading (rad/s) and θ is used to describe the LOS angle. Note, this angle is neither equal to our state θ which is defined in Eq. (2.2) nor it is the same as the pitch angle described in Section 2.3.3. The notation $\dot{\theta}$ to signify LOS rate is used so as to maintain consistency with the missile guidance literature. In order to understand how PN guidance law works, it is important to understand the concept of LOS rate. In the purview of our problem, LOS is defined as the imaginary line joining the UAV and its goal at any given instant of time. This line changes in length and orientation as the UAV maneuvers towards the goal. The change in its angular orientation is called the LOS rate and its given by the rate of turn, which is usually expressed in rad/s .

FIGURE 2.2: UAV-Goal Engagement Geometry

The equations of motion based on the UAV-goal geometry [92], are given below with reference to Figure. 2.2:

$$V_r = \dot{d}_g = -u \cos \alpha_g \quad (2.28)$$

$$V_\theta = d_g \dot{\theta} = u \sin \alpha_g \quad (2.29)$$

where, V_r is the rate of change of distance between the UAV and goal (i.e., the component of relative velocity along the LOS) and V_θ is the angular velocity of the LOS (i.e., the component of relative velocity perpendicular to the LOS), α_g and d_g are the estimated states of the UAV, obtained using Algorithm 1 and u is the forward velocity obtained from the vehicle's Inertial Measurement Unit (IMU). From Eq. (2.29), we can write:

$$\dot{\theta} = \frac{u \sin \alpha_g}{d_g} \quad (2.30)$$

Substituting Eq. (2.30) in Eq. (2.27), we get

$$r_c = \dot{\psi} = \frac{Nu \sin \alpha_g}{d_g} \quad (2.31)$$

u_c is a constant limited by vehicular and environmental constraints. In simulation, u_c varies from 0.1 to 0.5 m/s. Both the control inputs are used by the velocity controller described in Section 2.3.3 to generate autopilot control commands (F_c, ϕ_c, θ_c) . r_c is directly used by the autopilot.

2.3.2 Obstacle Avoidance Controller

In unstructured environments, robot motion is often associated with uncertainty as the environment may be dynamically changing. Even in static environments, the presence of other structures, humans or pets can pose the risk of collision as the UAV navigates towards its goal. More generally put, these can create occlusions due to which the UAV might not be able to receive direct visual measurements from the goal, resulting in it possibly getting lost or unable to complete its mission. Therefore, for a collision-free motion to the goal, the relative orientation based path planning (as described in Section 2.3.1) has to be associated with a local obstacle avoidance capability. For the purposes of our experiment, it is assumed that the obstacles themselves have no motion, i.e., the environment is static.

FIGURE 2.3: A conceptual view of the UAV approaching an obstacle enroute to its goal

The UAV used in our experiments is only equipped with a forward-pointing monocular camera, thus, it does not receive any depth information from the obstacles in its environment. It can detect obstacles or landmarks that pose as one, using the feature detection technique described in Section 2.4. Based on this information, the obstacle avoidance controller has to generate a trajectory that guarantees a collision-free path to the goal. Essentially, it works in tandem with the PN guidance controller based on the following rules or conditions:

- Until and unless the UAV is within the collision radius (r_{col}) of any obstacle, it follows the control commands generated by the PN guidance law.
- Even when the distance of the UAV from the obstacle, $d_{obs} < r_{col}$, it maybe possible that the obstacle is present on the very edge of its camera's FOV, and thus, not really occluding its path to the goal. As such, it is also necessary to impose a difference in bearing angle criterion that determines whether or not to activate the obstacle avoidance controller. Let, α_{obs} be the azimuth bearing angle of the UAV with respect to the obstacle and α_{diff} be the minimum bearing difference below which the goal is considered to have been occluded by the obstacle. Then, the control law switches from PN guidance to obstacle avoidance only if:

$$d_{obs} < r_{col} \quad \text{and} \quad (\alpha_{obs} - \alpha_g) < \alpha_{diff} \quad (2.32)$$

This is called the entry criterion for obstacle avoidance.

- The exit criterion is determined, again, using the difference in bearing angle which switches the controller back to PN guidance law once $(\alpha_{obs} - \alpha_g) \geq \alpha_{diff}$.

The obstacle avoidance control law maneuvers the UAV to push the obstacles to the edges of the camera FOV such that a constant bearing is maintained with them. This

strategy based on [70] avoids immediate collision with obstacles based on a deterministic reactive guidance approach.

Similar to Eqs. (2.28) and (2.29), the kinematic equations for the obstacle-UAV geometry can be written as:

$$\dot{d}_{obs} = -u \cos(\alpha_{obs}) \quad (2.33)$$

$$\dot{\theta}_{obs} = \frac{u \sin(\alpha_{obs})}{d_{obs}} \quad (2.34)$$

Here, θ_{obs} is LOS angle to the obstacle. We know that,

$$\alpha_{obs} = \theta_{obs} - \psi \quad (2.35)$$

Differentiating Eq. (2.35) and substituting the value of $\dot{\theta}_{obs}$ from Eq. (2.34), we get

$$r_c = \dot{\psi} = \frac{u \sin(\alpha_{obs})}{d_{obs}} - \dot{\alpha}_{obs} \quad (2.36)$$

$\dot{\alpha}_{obs}$ is the rate of change of the UAV's bearing with respect to the obstacle and can be expressed in terms of a proportional controller that maintains a fixed desired angle with the obstacle. Therefore,

$$r_c = \frac{u \sin(\alpha_{obs})}{d_{obs}} - k_p(\alpha_{obs} - \alpha_{obs_d}) \quad (2.37)$$

Here, α_{obs_d} is an angle less than the horizontal FOV of the UAV's camera. This ensures that the UAV traverses a path with an objective of keeping the obstacle α_{obs_d} radians away. It is necessary to keep the obstacle within the FOV until the exit criterion is reached because in the absence of any global positioning data, the UAV can only rely on its visual sensor in order to ensure that it does not collide with the obstacle. Therefore, before the entry criterion and after the exit criterion, the UAV changes its orientation based on PN guidance law (Eq. (2.31)) and within this zone, the UAV's heading is manipulated using Eq. (2.37).

2.3.3 Velocity Controller

In addition to the controllers described in Sections 2.3.1 and 2.3.2, a velocity controller was developed in the simulation environment in order to map the linear and angular velocities (u, v, w, p, q, r described below) to the thrust (F_c) and commanded angle or angular rates (ϕ_c, θ_c, r_c) that act as inputs to the UAV's autopilot. Here, ϕ_c and θ_c

are the commanded roll and pitch angles (in rad) respectively; r_c is the commanded yaw rate in rad/s. This makes the simulated model exactly the same as the UAV used for hardware experiments, which accepts only linear and angular velocity inputs. This is done so that a uniform control law can be formulated across both simulation and hardware platforms.

We'll begin the derivation by mentioning the states of a quadrotor UAV according to the 12-states model [93]:

p_n = the inertial North position of the UAV along i^i in $\mathcal{F}^i$

p_e = the inertial East position of the UAV along j^i in $\mathcal{F}^i$

p_d = the inertial Down position of the UAV along $-k^i$ in $\mathcal{F}^i$

u = UAV's body frame velocity measured along i^b in $\mathcal{F}^b$

v = UAV's body frame velocity measured along j^b in $\mathcal{F}^b$

w = UAV's body frame velocity measured along k^b in $\mathcal{F}^b$

ϕ = the roll angle defined with respect to $\mathcal{F}^{v2}$

θ = the pitch angle defined with respect to $\mathcal{F}^{v1}$

ψ = the yaw angle defined with respect to $\mathcal{F}^v$

p = the roll rate measured along $\mathcal{F}^b$

q = the pitch rate measured along $\mathcal{F}^b$

r = the yaw rate measured along $\mathcal{F}^b$

FIGURE 2.4: Definition of axes

The relation between each of the above-mentioned reference frames and a detailed derivation for the quadrotor kinematic and dynamic model are illustrated in [93] and as such, are excluded from this thesis for brevity. The rigid body dynamic equation which is essential to the derivation of our velocity controller is defined as (Eq. 17 from [93]):

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} rv - qw \\ pw - ru \\ qu - pv \end{bmatrix} + \begin{bmatrix} -g \sin \theta \\ g \cos \theta \sin \phi \\ g \cos \theta \cos \phi \end{bmatrix} + \frac{1}{m} \begin{bmatrix} 0 \\ 0 \\ -F \end{bmatrix} \quad (2.38)$$

where, g is the gravity which acts in $\mathcal{F}^i$ and m is the mass of the UAV (in kg). The body-frame accelerations can be written in terms of a proportional gain K multiplied with the respective errors in body-frame velocities. Eq. (2.38) can, therefore, be rewritten as following:

$$\begin{bmatrix} rv - qw \\ pw - ru \\ qu - pv \end{bmatrix} - K \begin{bmatrix} u - u_c \\ v - v_c \\ w - w_c \end{bmatrix} = \begin{bmatrix} -g \sin \theta \\ g \cos \theta \sin \phi \\ g \cos \theta \cos \phi \end{bmatrix} + \frac{1}{m} \begin{bmatrix} 0 \\ 0 \\ -F \end{bmatrix} \quad (2.39)$$

Here, u_c , v_c and w_c are the desired body-frame velocities (in m/s) acting along $\hat{i}^b$, $\hat{j}^b$ and $\hat{k}^b$ axis respectively. The value of K is obtained using a simple Linear Quadratic Regulator (LQR) which is detailed in Appendix 7.

Let,

$$\begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} = \begin{bmatrix} rv - qw \\ pw - ru \\ qu - pv \end{bmatrix} - K \begin{bmatrix} u - u_c \\ v - v_c \\ w - w_c \end{bmatrix} \quad (2.40)$$

From Eqs. (2.39) and (2.40), we can write:

$$u_x = -g \sin \theta \quad (2.41)$$

$$u_y = g \cos \theta \sin \phi \quad (2.42)$$

$$u_z = g \cos \theta \cos \phi - \frac{F}{m} \quad (2.43)$$

At $\theta = \theta_c$, we can write Eq. (2.41) as:

$$\theta_c = \sin^{-1}\left(\frac{-u_x}{g}\right) \quad (2.44)$$

Similarly at $\phi = \phi_c$, Eq. (2.42) can be written as:

$$\phi_c = \sin^{-1}\left(\frac{u_y}{g \cos \theta}\right) \quad (2.45)$$

Finally, from Eq. (2.43), commanded thrust F_c can be denoted as:

$$F_c = m(g \cos \theta \cos \phi - u_z) \quad (2.46)$$

The higher-level path planning algorithm, such as the ones described in Sections 2.3.1 and 2.3.2, generates u_c, v_c, w_c (Eq. (2.39)) and the commanded roll, pitch and yaw rates (p_c, q_c, r_c) that act as inputs to the velocity controller. In order to reduce the complexity of our EKF state modeling (refer to Eq. (2.12)), we restrict the UAV to move in a 2D plane (similar to a ground robot). As a result, v_c, w_c, p_c, q_c are all set to zero and the UAV only has a commanded forward velocity, u_c and an angular turn rate along $\hat{k}^b$ axis, r_c .

2.4 Visual Tracking

As mentioned in Chapter 1, the objective of the UAV is to autonomously fly to the desired goal location. The goal is selected by the user from the images provided by its onboard camera. The goal's relative orientation with respect to the UAV is estimated using an Extended Kalman Filter (EKF) to combine the information from the motion of the vehicle and the measurement provided by the camera bearing sensor. The UAV also tracks the evolution of its orientation with respect to other points of interest in the environment, called Landmarks (LMs). These help in improving its estimate of the goal's position. In this section, we discuss the various computer vision techniques used to detect and track the goal and LMs in the UAV's image frame as it flies towards the goal. The computer vision techniques can, essentially, be categorized into:

- Goal Selection
- Goal Tracking
- Landmark Detection
- Landmark Tracking

2.4.1 Goal Selection

The goal is selected by the user/operator clicking on the image with a mouse-pointer. Before switching to autonomous flight mode where the UAV receives active control commands, it is taken off remotely by the operator and made to hover at an altitude so that the desired goal is within its camera's FOV. The user, then, selects the four corners

of an object or the goal as shown in Figure. 2.5. This constitutes a template image, which is, used to track the goal, as described in Section 2.4.2.

FIGURE 2.5: Goal Template Selection

The top-left corner of the image frame acts as the origin and the pixel location of any object in the frame, (x_i, y_i) , is defined by its position in the (i_{im}, j_{im}) coordinate frame. Goal selection, as proposed in [54], involves choosing the center of the object and then defining a template around it based on a fixed predetermined template width and height, dxT and dyT (in pixels) respectively. This, however, gives less flexibility as to what the shape and size of the desired goal can be. This is illustrated in Figure. 2.6 where a 70x70 template is sufficient to track the cardboard box, however, the same template barely covers if the door were to be selected as the goal in the image.

FIGURE 2.6: Predetermined template size cannot account for different goal sizes

Additionally, if dxT and dyT are very small, due to increasing object size as the UAV approaches it, the goal's dimensions can quickly exceed that of the template, causing it to possibly go untracked in the subsequent image frames. This is called template drift, which will occur despite our choice of goal selection method. However, allowing the user to select the template corners instead of its center provides dynamic height and width allocation as shown in Figure. 2.7. This allows them to choose a wide range of objects and/or keep a buffer zone within the template window such that the goal can be tracked for a longer period of time without having to offset the effects of template drift. In this case, the center of the template gives the goal's pixel coordinates, which is denoted by (x_g, y_g) in Figure. 2.5. These are used in Eq. (2.22) to calculate the azimuth and elevation bearing angle of the goal with respect to the UAV.

FIGURE 2.7: Template based on goal size

2.4.2 Goal Tracking

Once the template has been created, the next objective is to find the goal in each subsequent image frame provided by the UAV's camera. This is achieved using a technique called template matching [94, 95]. It works by sliding the template (defined by its width dxT and height dyT pixels respectively) over the entire image frame, defined by its width W pixels and height H pixels. Sliding means moving the template one pixel at a time from left to right and then, from top to bottom. At each location, a metric is calculated which represents how similar the template is to that particular area of the source image. For each location of the template T over the image frame I , the metric is stored in a result matrix R , each (x, y) coordinate of which contains the similarity score at that location. The similarity score is a number between 0 and 1, with 1 being the highest match possible.

The template matching algorithm generates the following outputs:

- The location with the highest match, (x_T, y_T) and its similarity score, S_{high}
- The location with the least match, (x_L, y_L) and its similarity score, S_{low}
- R - the result matrix of size $(W - dxT + 1, H - dyT + 1)$

(x_T, y_T) actually represents the location of the source image which has the highest match with the top-left corner of the template image. Therefore, the position of the center of the template can be calculated as:

$$\begin{bmatrix} x_g \\ y_g \end{bmatrix} = \begin{bmatrix} x_T + dxT/2 \\ y_T + dyT/2 \end{bmatrix} \quad (2.47)$$

As the UAV moves, so does the position of the goal in the image frame. Thus, (x_T, y_T) and consequently, (x_g, y_g) changes based on the highest matching or similarity coefficient. The tracking algorithm is developed so as to only accept matches above a certain similarity threshold, S_{thresh} . Lower values of S_{thresh} can result in false positives (detecting the goal when it's actually not in the frame) or true negatives (not tracking the goal or tracking a completely different object even though the goal is in the frame). The latter case can be seen in Figure. 2.8. Setting a very high S_{thresh} , on the other hand, can result in the goal being untracked even for slightest changes in the template.

FIGURE 2.8: Template size increases as UAV approaches the goal

2.4.2.1 Template Drift

As mentioned in Section 2.4.1, goal tracking poses the problem of template drift. Therefore, in order to accurately obtain measurements from the goal throughout the mission,

it is necessary to update the size of its template as and when the size of the object increases in the image frame, as shown in Figure. 2.9. It can be seen that the size of the door, which is selected as the goal in this case, is significantly larger in Figure. 2.8 when the goal is nearer than at the beginning in Figure. 2.7. It is desired that the template size should be big enough to accommodate the entire object and small enough so that the template matching algorithm can run efficiently.

FIGURE 2.9: Template size increases as UAV approaches the goal

Template drift is compensated by increasing the size of the template when the area of the largest contour within it exceeds a certain percentage of the template area. This is done by passing the template image (T) through a Hue-Saturation-Value (HSV) filter. At the time of goal selection, the maximum and minimum HSV values of T are saved so as to find pixels that fall within these limits. The filter is only used over the template image because the goal can have similar HSV values to other objects present in the scene, such as the boxes in Figure. 2.7. The image is passed through morphological operators [96] to remove the noises produced by thresholding. The algorithm then finds the largest contour within this HSV range, which is most likely to be that of the goal. During the mission, whenever the area of this contour becomes greater than 50% of the current template area, the template size is increased by a certain percentage. T is updated so that whenever the next template matching occurs, the newer, larger template is slid over the source image, instead of the previous smaller one.

2.4.2.2 Goal Reacquisition

Even though the UAV starts the mission with the goal in its FOV, there can be times when it is not seen by the camera. This is possible when (i) the obstacle gets in the path of the UAV and occludes its view of the goal (Figure. 2.10), or (ii) the UAV switches

its path based on the commands of the Obstacle Avoidance Controller in a manner that the goal moves out of its FOV (Figure. 2.11).

FIGURE 2.10: Obstacle in the UAV's Direct Line-of-Sight to Goal

FIGURE 2.11: Goal leaves FOV while UAV tries to avoid obstacle

In such situations, it is important to recognize and re-track the goal once it is back in the camera's FOV. The template matching technique described before is able to do so efficiently because in every frame, it saves a copy of the template patch which shows how the goal looked like before it went untracked.

2.4.3 Landmark Detection

Landmark Detection is achieved using OpenCV's *goodFeaturesToTrack* function [97]. The function finds n most prominent corners in the image or in the specified image region using Shi-Tomasi Corner Detector and returns a corner quality measure at every source image pixel [73]. These corners act as Landmarks (LMs) that help localize the

UAV with respect to its surroundings. The user provides a corner quality level, Q_{corner} (usually between 0 and 1), which specifies the minimal accepted quality of image corners. The corners with a quality level less than $(Q_{corner} \times Q_{Best})$ are rejected and hence, don't contribute to the state vector defined in Eq. (2.2). Here, Q_{Best} is the quality level of the most prominent corner in the scene. Also, if two detected corners are within d_{corner} pixels of each other, they are said to be the same feature or landmark and only one of them is used for calculating the camera bearing measurements for that particular landmark. The EKF also has limits regarding the number of LMs that can be added to the state vector. Once this number n_{max} is reached, no more features are added into the state vector. This feature detection technique is used to initialize a point-based tracker (Section 2.4.4) which follows the change in LM position due to the UAV's motion.

FIGURE 2.12: Landmarks and their surrounding templates

Figure. 2.12 shows three detected LMs and their surrounding templates of designated width and height, dx_i and dy_i pixels respectively. As mentioned in Chapter 1, our problem formulation does not involve developing a dense, volumetric map of the environment. It's only concerned with the evolution of the UAV's states with respect to fixed points or features in the environment. Therefore, while it is an absolute requirement to visually track the goal throughout the mission in order to successfully navigate to it, ability to see the entire object in order to track a LM is not a necessity. As long as the detected corners can be associated correctly during each measurement, they can help in improving the UAV's estimates as a whole. Therefore, it is sufficient to provide the LM tracker with fixed template dimensions unlike in the case of goal tracking.

2.4.4 Landmark Tracking

Once a set of LMs have been detected, they are tracked in every new frame using a centroid tracker, as described in [98]. Here, centroid refers to the center of a LM's template which is basically the corner detected by the *goodFeaturesToTrack* function. Therefore, we can write

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} x_{T_i} + dx_i/2 \\ y_{T_i} + dy_i/2 \end{bmatrix} \quad (2.48)$$

where, (x_{T_i}, y_{T_i}) refers to the top-left corner of the i^{th} LM's template. When no objects are being tracked, the algorithm loops through all the detected corners and creates a list of LMs, each with its unique ID. After that, for every new image frame, a distance matrix, D , is calculated which stores the distance between every pair of existing LMs and the new corners received in that frame. Let, the existing LMs and the new corners be denoted by LM_{old_i} and C_{new_i} . For a single iteration of the tracker, let us assume the following sets of values:

- $LM_{old_1} = (175, 105)$, $LM_{old_2} = (345, 175)$, $LM_{old_3} = (398, 51)$
- $C_{new_1} = (320, 191)$, $C_{new_2} = (200, 85)$, $C_{new_3} = (335, 10)$, $C_{new_4} = (118, 206)$

i.e., there are 3 existing LMs and in the current frame, 4 corners have been detected, as shown in Figure. 2.13.

FIGURE 2.13: Pairwise distance calculation between existing LMs and new corners

Then, the matrix D can be written as:

$LM_{old_i} \backslash C_{new_i}$	(320, 191)	(200, 85)	(335, 10)	(118, 206)
(175, 105)	168.59	32.02	186.08	115.97
(345, 175)	29.68	170.66	165.30	229.11
(398, 51)	160.26	200.89	75.17	320.04

TABLE 2.1: Distance between existing LMs and new corners (in pixels)

The algorithm now finds the minimum distance in each row of D matrix. The column in which this minimum distance occurs gives the new corner which is closest to the existing LM, and therefore, it can be defined as the new pixel location of that particular LM in the current frame. For example, the minimum value in the first row is 32.02 which signifies that the landmark which was at (175, 105) pixel in the previous frame is at (200, 85) pixel in the current frame. For every frame, the updated LM locations are used to calculate the bearing measurements using Eq. (2.23). If the minimum distance in any row exceeds d_{max} pixels, then the C_{new_i} for that column is considered to be a spurious detection and that corner is rejected. For instance, if d_{max} is 60 pixels, then the C_{new_i} corresponding to the value minimum value in row 3 (75.17 pixels), i.e., (335, 10) is discarded. This implies that for this image frame, there is no bearing measurement available for the third landmark. Also, it can be observed from Table 2.1 that the number of new corners is greater than the number of existing landmarks. In such cases, there are two possibilities:

- One of the previously detected LMs have disappeared from the camera's FOV. If a landmark has not been matched to any new corners for up to F_{max} frames, it is considered to be no longer visible to the UAV. The bearing sensor returns no measurement to the EKF for such a landmark.
- A new LM has been detected at the position corresponding to C_{new_4} and it is registered with a new ID that follows the last assigned ID. Therefore, if the landmarks at positions (175, 105), (345, 175) and (398, 51) has IDs 0, 1, 2 associated with them respectively, then C_{new_4} is added to the list along with its ID 3.

2.4.4.1 Data Association

Data Association in terms of landmark tracking can be defined as the ability of the UAV to match observed landmarks from different measurements with each other. Simply put,

it is the ability to recognize whether a particular landmark has been viewed by the UAV before. For example, in Figure. 2.14, the unique ID 0 should associate with the same landmark in both the left and right-hand side frames representing frame numbers 51 and 105 respectively.

FIGURE 2.14: Landmark Association over multiple image frames

Wrongly associating a new landmark with a previously seen landmark can be devastating as it'll produce erroneous measurements, causing the UAV to think it is at a different location than it actually is. This is represented with the same two frames in Figure. 2.15. In this case, the new feature detected in frame 105 is incorrectly associated as the same landmark with ID 0 in frame 51. As we can see, they are two completely different objects present far apart from each other in the scene and hence will have dissimilar relative orientation with respect to the UAV. Such incorrectly associated LMs should be filtered out in order to prevent the state estimates from deteriorating. This is achieved using the centroid tracker mentioned in Section 2.4.4, which rejects any new corners that are beyond d_{max} pixels away from any of the existing LMs.

FIGURE 2.15: Incorrect Landmark Association

Other possible problems that can arise during data association are:(i) The landmarks might not be re-observed at every time step (ii) Some corners may be identified as LMs but the tracker may fail to see it again. Such LMs are not suitable as reference points in the environment and are, again, rejected or discarded from the measurement vector after being untracked for a maximum of F_{max} frames.

2.5 Summary

In this Chapter, we provide a detailed description of our problem formulation, starting with the definition of the system's states, state estimation, control design and finally, the computer vision tools that are used in this project. We derive the motion and measurement model along with their respective Jacobians which are used by the EKF's predict and update steps respectively. The output of the EKF goes to a two-step controller which sends a forward linear velocity and yaw rate command to the UAV. The controller switches between PN guidance and obstacle avoidance based on some entry and exit criteria, as described in Section 2.3.2. Section 2.4 illustrates the goal selection and tracking, landmark detection and tracking techniques used in this work. Real-time images collected during experiments are used to explain how the tracker works. Finally, this section also covers some challenges posed by such tracking problems and how they are mitigated.

Chapter 3

Observability Analysis

In control theory, observability is defined as the measure of how well the internal states of a system can be inferred based on the information provided by its external outputs [99]. A system is said to be observable if the current value of its state variables can be determined by observing its sensor measurements for a finite period of time. This can be explained simply using the following examples -

- A robot moving in a circular trajectory using range-only measurements from a way-point (WP) at the center of the circle is an unobservable system. This is because at every point on its trajectory, the robot receives the same range measurement from the WP at the center, as shown in Figure. 3.1a. As a result, the robot thinks that it can be anywhere on this circular path because it cannot distinguish between any two sets of measurements.

FIGURE 3.1: Cases when a system may become unobservable- (a) Circular Trajectory with range-only measurement, (b) Straight-line path with bearing-only measurement

- Similarly, in a bearing-only measurement case, if a robot moves in a straight-line path towards a WP from which it receives measurements, the system becomes

unobservable. This is again due to the fact that two sets of measurements are indistinguishable, thus, the robot effectively has no information regarding the change in its states. This is illustrated in Figure. 3.1b.

The second case also makes the controller design for this work more intuitive - a simple proportional controller will quickly cause the bearing angle to converge resulting in indistinguishable measurements and hence, little information regarding the system's current states. The observability of this system is derived based on the nonlinear observability analysis developed in [100]. Section 3.1 briefly describes this concept and then analyzes the observability of the system for the following three cases:

- The UAV can only see the goal and no Landmarks (LMs) in its Field-of-View (FOV) (Section 3.2).
- The UAV observes the goal and m LMs. Here, $m \in n$ (the total number of LMs) and the objective is to find the minimum number of LMs required for the system to be observable (Section 3.3).
- The UAV can see only the minimum number of LMs but not the goal (Section 3.4).

The conclusions from observability analysis are, then, compared with the goal's $3 - \sigma$ plots from simulation. The simulation parameters are mentioned in Table 5.1. A system that is observable has sufficient information about its states such that the uncertainty associated with their respective beliefs reduces over time, i.e., their $3 - \sigma$ plots converge. On the other hand, in an unobservable system, the error in estimation continuously increases because sufficient measurements are not available to improve the system's beliefs regarding its states.

This work assumes 2D motion of the UAV while designing the controller, as a result of which, while deriving the system motion model in Section 2.2.1, we had assumed $v = w = 0$. The observability analysis presented in this Chapter also considers a 2D system as this is significantly less computation-intensive than the 3D case. However, it is important to note that this analysis doesn't involve any system linearization. The inherent nonlinearity of the system is still maintained in the 2D case, thus, providing enough information to shed light on the system's observability, both mathematically and intuitively. For a 2D system, we can also consider $p = q = 0$. This assumption was not made in motion model in order to incorporate the process noises due to any sudden motion of the UAV. Additionally, the elevation angle of the UAV w.r.t the goal and LMs also become zero, i.e., $\beta_g = \eta_i = 0$.

3.1 Lie Derivatives

Let, $f : \mathbb{R}^N \rightarrow \mathbb{R}^N$ be a vector field in $\mathbb{R}^N$ and $h : \mathbb{R}^N \rightarrow \mathbb{R}$ be a smooth scalar function. Then the Lie derivative of h with respect to f is given by:

$$L_f h = (\nabla h)f = \left(\frac{\partial h}{\partial X}\right)f \quad (3.1)$$

$\frac{\partial h}{\partial X}$ is basically the measurement model Jacobian, H , described in Section 2.2.3. However, depending on the condition we are analyzing the system's observability for, h and hence, H will change based on the number of available sensor measurements. f in Eq. (3.1) refers to the combined goal and landmark motion model defined in Eqs. (2.17) and (2.19) respectively. Lie derivatives are recursive which basically means that a higher order Lie derivative can be calculated by finding the partial differential of its previous order Lie derivative w.r.t X . Zeroth-order Lie derivative of a function h is denoted by the function h itself, i.e.,

$$L_f^0 h = h$$

It gives the information that can be inferred about the states X based on the sensor input h under no motion of the system. First-order Lie derivative is used to calculate the information h gives about X under first-order motion using the function f . Based on the recursive property, the first-order Lie derivative can be calculated as:

$$L_f^1 h = (\nabla L_f^0 h)f = (\nabla h)f$$

Similarly,

$$L_f^2 h = (\nabla L_f^1 h)f$$

and,

$$L_f^{N-1} h = (\nabla L_f^{N-2} h)f$$

$$L_f^N h = (\nabla L_f^{N-1} h)f$$

Let,

$$l(X) = \begin{bmatrix} L_f^0 h & L_f^1 h & \cdots & L_f^{N-1} h \end{bmatrix}^T \quad (3.2)$$

Then, the observability matrix $\mathcal{O}$ is calculated as:

$$\mathcal{O} = \frac{\partial l(X)}{\partial X} \Big|_{X=X_0} \quad (3.3)$$

For a system with N states, Lie derivatives up to at least $(N - 1)^{th}$ order should be calculated in order to ascertain the rank of its observability matrix. If $\mathcal{O}$ has N linearly independent rows, then the system is said to be locally weakly observable at X_0 . We know that the Row Reduced Echelon Form (RREF) is the simplest possible form of a matrix, which directly gives the number of linearly independent rows. Therefore, at first, $\mathcal{O}_r = RREF(\mathcal{O})$ is calculated and then, using [101], an elementary operation matrix, E is calculated as:

$$E = \frac{\mathcal{O}_r}{\mathcal{O}} \quad (3.4)$$

The conditions for which a system becomes unobservable can be obtained intuitively based on the elements in the linearly independent rows of E . This will become more clear as the observability conditions are derived for the 3 cases mentioned earlier. However, before proceeding to these cases, Lie derivative calculation is explained using a simple example as follows:

Example

$$h(x, y) = x - y^2, \quad g(x, y) = \begin{bmatrix} x & y \end{bmatrix}^T, \quad f(x, y) = \begin{bmatrix} y & x^2 \end{bmatrix}^T$$

Calculate $L_g L_f H$.

$$\begin{aligned} L_g L_f H &= \nabla(L_f h)g \\ &= \nabla(\nabla h f)g \\ &= \nabla\left(\frac{\partial h}{\partial X} f\right)g \end{aligned}$$

Here, $X = \begin{bmatrix} x & y \end{bmatrix}^T$. Therefore,

$$\begin{aligned} L_g L_f H &= \nabla\left(\begin{bmatrix} \frac{\partial h}{\partial x} & \frac{\partial h}{\partial y} \end{bmatrix} \begin{bmatrix} y \\ x^2 \end{bmatrix}\right) g \\ &= \nabla\left(\begin{bmatrix} 1 - y^2 & x - 2y \end{bmatrix} \begin{bmatrix} y \\ x^2 \end{bmatrix}\right) g \\ &= \nabla(y - y^3 + x^3 - 2x^2 y) g \\ &= \frac{\partial}{\partial X}(y - y^3 + x^3 - 2x^2 y) \begin{bmatrix} x \\ y \end{bmatrix} \end{aligned}$$

$$\begin{aligned}
&= \begin{bmatrix} 2x^2 - 4xy & 1 - 3y^2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \\
&= 2x^3 - 4x^2y + y - 3y^3
\end{aligned}$$

3.2 Case I: Goal Observability

For analyzing this case, we consider a scenario with no obstacles along the UAV's path so that it can get measurement from the goal at all points of time. The UAV does not receive measurements from any LMs in this case. The state and measurement vectors can, then, be defined as:

$$X = \begin{bmatrix} \alpha_g & d_g \end{bmatrix}^T, \quad h = \begin{bmatrix} \alpha_g \end{bmatrix} \quad (3.5)$$

Considering a 2D case, $f(X, U)$ can be written as:

$$f(X, U) = \begin{bmatrix} \dot{\alpha}_g \\ \dot{d}_g \end{bmatrix} = \begin{bmatrix} u \sin \alpha_g / d_g \\ -u \cos \alpha_g \end{bmatrix} \quad (3.6)$$

This system is locally weakly observable if its observability matrix, $\mathcal{O}$, is full rank, i.e., $\text{rank}(\mathcal{O}) = 2$. The zeroth and first-order Lie derivatives of the system need are calculated as follows:

$$\begin{aligned}
L_f^0 h &= h = \begin{bmatrix} \alpha_g \end{bmatrix} \\
L_f^1 h &= (\nabla h) f = \begin{bmatrix} \frac{\partial \alpha_g}{\partial \alpha_g} & \frac{\partial \alpha_g}{\partial d_g} \end{bmatrix} \begin{bmatrix} u \sin \alpha_g / d_g \\ -u \cos \alpha_g \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} u \sin \alpha_g / d_g \\ -u \cos \alpha_g \end{bmatrix} \\
&= \frac{u \sin \alpha_g}{d_g}
\end{aligned}$$

Using Eqs. (3.2) and (3.3), $\mathcal{O}$ can be written as:

$$\begin{aligned}
\mathcal{O} &= \begin{bmatrix} 1 & 0 \\ \frac{u \cos \alpha_g}{d_g} & \frac{-u \sin \alpha_g}{d_g^2} \end{bmatrix} \\
\mathcal{O}_r &= RREF(\mathcal{O}) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}
\end{aligned}$$

FIGURE 3.2: Case I - Initial position of UAV and goal

FIGURE 3.3: Case I - $3-\sigma$ Plots - α_g and d_g

Since the two rows of $\mathcal{O}_r$ are linearly independent, it shows that the system is observable for goal-only case. This is also illustrated in Figure. 3.3 where the $3-\sigma$ plots of α_g and d_g converge over time. It assumes the initial position of the UAV and goal as shown in Figure. 3.2. Using Eq. (3.4), E can be calculated as:

$$E = \begin{bmatrix} 1 & 0 \\ \frac{d_g \cos \alpha_g}{\sin \alpha_g} & \frac{-d_g^2}{u \sin \alpha_g} \end{bmatrix}$$

Analyzing $E(2, 1)$ and $E(2, 2)$, the sufficient conditions for system observability can be stated as follows:

- $d_g \neq 0$. It basically means that the UAV and goal cannot be located at the same place. Then the 2nd row of E becomes 0 and the system is no longer full rank.
- $u \neq 0$. When the UAV is stationary, i.e., $u = 0$, whether it is on ground or hovering at a fixed location, then its measurement w.r.t the goal does not change. Therefore, there is no way to triangulate its position if there is no motion.
- $\alpha_g \neq 0, k\pi$. If the vehicle moves along the vector joining it to the goal, then $\alpha_g = 0, k\pi$. This provides no new information about the system. This is same as the case described in Figure. 3.1a.

3.3 Case II: Goal and Landmark Observability

The objective of this case is to find out the minimum number LMs that the UAV needs to observe in addition to the goal for the system to be observable. At first, a single LM is considered. The state and measurement vector and the system motion model can, then, be written as:

$$X = [\alpha_g \ d_g \ \theta_1 \ d_1]^T, \quad h = [\alpha_g \ \gamma_1]^T \quad (3.7)$$

$$f(X, U) = \begin{bmatrix} \dot{\alpha}_g \\ \dot{d}_g \\ \dot{\theta}_1 \\ \dot{d}_1 \end{bmatrix} = \begin{bmatrix} u \sin \alpha_g / d_g \\ -u \cos \alpha_g \\ \frac{u \sin(\alpha_g + \theta_1)}{d_1} - \frac{u \sin \alpha_g}{d_g} \\ -u \cos(\alpha_g + \theta_1) \end{bmatrix} \quad (3.8)$$

The zeroth and first-order Lie derivatives of the system can be calculated as:

$$\begin{aligned} L_f^0 h &= h = [\alpha_g \ \gamma_1]^T \\ L_f^1 h &= (\nabla h) f = \begin{bmatrix} \frac{\partial \alpha_g}{\partial \alpha_g} & \frac{\partial \alpha_g}{\partial d_g} & \frac{\partial \alpha_g}{\partial \theta_1} & \frac{\partial \alpha_g}{\partial d_1} \\ \frac{\partial \gamma_1}{\partial \alpha_g} & \frac{\partial \gamma_1}{\partial d_g} & \frac{\partial \gamma_1}{\partial \theta_1} & \frac{\partial \gamma_1}{\partial d_1} \end{bmatrix} f \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{u \sin \alpha_g}{d_g} \\ -u \cos \alpha_g \\ \frac{u \sin(\alpha_g + \theta_1)}{d_1} - \frac{u \sin \alpha_g}{d_g} \\ -u \cos(\alpha_g + \theta_1) \end{bmatrix} \end{aligned}$$

$$= \begin{bmatrix} \frac{u \sin \alpha_g}{d_g} \\ \frac{u \sin(\alpha_g + \theta_1)}{d_1} \end{bmatrix}$$

Since this system has 4 states, we need to calculate at least till the third-order derivative. The observability matrix, $\mathcal{O}$, can be written by stacking the Jacobians of $L_f^0 h$, $L_f^1 h$, $L_f^2 h$ and $L_f^3 h$ as follows:

$$\mathcal{O} = \begin{bmatrix} \frac{\partial L_f^0 h}{\partial X} & \frac{\partial L_f^1 h}{\partial X} & \frac{\partial L_f^2 h}{\partial X} & \frac{\partial L_f^3 h}{\partial X} \end{bmatrix}^T \quad (3.9)$$

where,

$$\frac{\partial L_f^0 h}{\partial X} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

$$\frac{\partial L_f^1 h}{\partial X} = \begin{bmatrix} \frac{u \cos \alpha_g}{d_g} & \frac{-u \sin \alpha_g}{d_g^2} & 0 & 0 \\ \frac{u \cos(\alpha_g + \theta_1)}{d_1} & 0 & \frac{u \cos(\alpha_g + \theta_1)}{d_1} & \frac{-u \sin(\alpha_g + \theta_1)}{d_1^2} \end{bmatrix}$$

The last two derivatives include multiple terms within more than one element and hence, are not mentioned here for brevity. Furthermore, $\frac{\partial L_f^2 h}{\partial X}$ and $\frac{\partial L_f^3 h}{\partial X}$ do not contribute to the observability matrix. This means that $\text{rank}(\mathcal{O}_r) = 4$ with or without these terms. Therefore, we can write $\mathcal{O}_r$ simply by ignoring the last two terms as follows:

$$\mathcal{O} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ \frac{u \cos \alpha_g}{d_g} & \frac{-u \sin \alpha_g}{d_g^2} & 0 & 0 \\ \frac{u \cos(\alpha_g + \theta_1)}{d_1} & 0 & \frac{u \cos(\alpha_g + \theta_1)}{d_1} & \frac{-u \sin(\alpha_g + \theta_1)}{d_1^2} \end{bmatrix}$$

$$\mathcal{O}_r = \text{RREF}(\mathcal{O}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Since $\text{rank}(\mathcal{O}_r)$ is the same as the number of states of this particular system, the system is observable with goal and just one LM. This result is also validated in simulation using the initial conditions shown in Figure. 3.4. It should be noted that the positions of goal

and LM_1 are fixed whereas the position of the UAV changes based on the controller input, as described in Section 2.3.1. It can be seen from Figure. 3.5 that the $3 - \sigma$ plots for both the goal states converge over time signifying that the system has enough information about the goal based on the sensor inputs.

FIGURE 3.4: Case II - Initial position of UAV, goal and LM_1

FIGURE 3.5: Case II - $3 - \sigma$ Plots - α_g and d_g

Using Eq. (3.4), E can be written as:

$$E = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \frac{d_g \cos \alpha_g}{\sin \alpha_g} & 0 & \frac{-d_g^2}{u \sin \alpha_g} & 0 \\ -1 & 1 & 0 & 0 \\ 0 & \frac{d_1 \cos(\alpha_g + \theta_1)}{\sin(\alpha_g + \theta_1)} & 0 & \frac{-d_1^2}{u \sin(\alpha_g + \theta_1)} \end{bmatrix}$$

The first and third rows of E are linearly independent of each other. The values of $E(2, 1)$, $E(2, 3)$, $E(4, 2)$ and $E(4, 4)$ should be such that neither the second nor the fourth row consists of all zeros (or infinite). Based on this, we can derive the conditions for system observability. These conditions are listed as follows:

- $d_g \neq 0$, $d_1 \neq 0$. The UAV cannot be located either at the goal or at LM_1 .
- $u \neq 0$
- $\alpha_g \neq 0, k\pi$
- $\theta_1 \neq -\alpha_g$. If they are equal, i.e., then the denominators in row 4 become zero.

This shows that the conditions for goal state observability remain the same and each additional LM will also be observable as long as the UAV is not travelling directly along the vector joining itself to that LM. These results are consistent with the 3D observability analysis derived in [54]. It can also be concluded from this analysis that the system will be observable as long as the UAV receives measurements from the goal or goal and at least one landmark at all points of time. However, it should be noted that this may not always be possible due to presence of obstacles and/or noises in sensor measurements. As a result, it is important that the environment has other LMs w.r.t which the UAV can localize itself in case the goal or LM_1 leaves the FOV.

3.4 Case III: Landmark-only Observability

In this Section, we consider a case where the goal is present but the UAV receives no measurement from it. Considering a single LM system, the state and measurement vectors can be written as:

$$X = \begin{bmatrix} \alpha_g & d_g & \theta_1 & d_1 \end{bmatrix}^T, \quad h = \begin{bmatrix} \gamma_1 \end{bmatrix}^T \quad (3.10)$$

$f(X, U)$ is same as in Eq. (3.8). Again, the zeroth and first-order Lie derivatives can be calculated as:

$$\begin{aligned}
 L_f^0 h &= h = \begin{bmatrix} \gamma_1 \end{bmatrix} \\
 L_f^1 h &= (\nabla h) f = \begin{bmatrix} \frac{\partial \gamma_1}{\partial \alpha_g} & \frac{\partial \gamma_1}{\partial d_g} & \frac{\partial \gamma_1}{\partial \theta_1} & \frac{\partial \gamma_1}{\partial d_1} \end{bmatrix} f \\
 &= \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{u \sin \alpha_g}{d_g} \\ -u \cos \alpha_g \\ \frac{u \sin(\alpha_g + \theta_1)}{d_1} - \frac{u \sin \alpha_g}{d_g} \\ -u \cos(\alpha_g + \theta_1) \end{bmatrix} \\
 &= \frac{u \sin(\alpha_g + \theta_1)}{d_1}
 \end{aligned}$$

$L_f^2 h$ and $L_f^3 h$ are again not mentioned here due to brevity but they can easily be calculated in MATLAB based on their recursive property. Similar to Eq. (3.9), the observability matrix for this case can be written by stacking the Jacobians from zeroth to third-order derivative. Using MATLAB,

$$\mathcal{O}_r = RREF(\mathcal{O}) = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$rank(\mathcal{O}) = rank(\mathcal{O}_r) = 2$ which is less than the number of states in this case, i.e., 4. Thus, this system is unobservable. This case is then analyzed for 2 LMs, i.e., the system will now have 6 states and 2 measurements as depicted below:

$$X = \begin{bmatrix} \alpha_g & d_g & \theta_1 & d_1 & \theta_2 & d_2 \end{bmatrix}^T, \quad h = \begin{bmatrix} \gamma_1 & \gamma_2 \end{bmatrix}^T \quad (3.11)$$

Calculating up to the fifth-order Lie derivative, stacking their Jacobians, we find the rank of this observability matrix to be 4. Thus, the system containing the goal and 2 LMs such that the UAV receives measurements only from the two LMs, is unobservable as well. Similarly, the observability matrix of the system with 3 LMs was also found to be rank-deficient. Thus, it can be concluded with relative certainty that no matter the number of LMs in the system, as long as the UAV does not receive any measurements from the goal, the system remains unobservable.

The simulation results are shown using two LMs, the initial position of which along with the UAV and the goal are shown in Figure. 3.6. It can be seen from Figure. 3.7 that the $3 - \sigma$ bounds of α_g keep diverging from the very beginning as the UAV receives no measurements from the goal. The error in d_g also keeps increasing and sufficient information is not available to improve its high uncertainty.

FIGURE 3.6: Case III - Initial position of UAV, goal and two LMs

FIGURE 3.7: Case III - $3 - \sigma$ Plots - α_g and d_g

The trace of the observability Gramian matrix ($\mathcal{O}^T \mathcal{O}$) provides information regarding the overall observability of the system [102]. Thus, for an unobservable system such as the one presented in this case, we can find the total information provided by the measurements regarding the states of the system. $\mathcal{O}^T \mathcal{O}$ is a square matrix, hence, its trace can be calculated as the summation of the main diagonal elements.

$$tr(\mathcal{O}^T \mathcal{O}) = \frac{2u^2 \cos^2 \gamma_i}{d_i^2} + \frac{u^2 \sin^2 \gamma_i}{d_i^4} + 2 \quad (3.12)$$

where, $\gamma_i = \alpha_g + \theta_i$, as described in Eq. (2.1).

The formulation in [53] represents the LM states using the overall bearing angles, γ_i and λ_i , instead of θ_i and η_i . It refers to γ_i and λ_i as α_i and β_i respectively, however, for all practical purposes, they are one and the same. In this analysis, the notations γ_i and λ_i are used to maintain consistency for comparison. There is no specific 'goal' assignment in the environment. Thus, any waypoint that the robot has to navigate to, i.e., the goal, is basically mapped similar to any other landmarks in the environment. The 2D states of this can be written as:

$$X = \begin{bmatrix} \alpha_g & d_g & \gamma_1 & d_1 \end{bmatrix}^T, \quad h = \begin{bmatrix} \gamma_1 \end{bmatrix}^T \quad (3.13)$$

Based on Eq. (6) of [53], the 2D motion model of the system for single goal and LM can be written as:

$$f(X, U) = \begin{bmatrix} \dot{\alpha}_g \\ \dot{d}_g \\ \dot{\gamma}_1 \\ \dot{d}_1 \end{bmatrix} = \begin{bmatrix} u \sin \alpha_g / d_g - \omega_r \\ -u \cos \alpha_g \\ u \sin \gamma_1 / d_1 - \omega_r \\ -u \cos \gamma_1 \end{bmatrix} \quad (3.14)$$

It can be seen that this is similar to the 2D motion model represented by Eq. 3.8 except for an additional term ω_r which denotes the yaw rate of the robot. The Lie derivatives are calculated similar to the three cases mentioned before. Using MATLAB, the trace of the observability Gramian for the system using the formulation presented in [53] can be written as:

$$tr(\mathcal{O}^T \mathcal{O}) = \frac{u^2 \cos^2 \gamma_i}{d_i^2} + \frac{u^2 \sin^2 \gamma_i}{d_i^4} + 1 \quad (3.15)$$

Comparing Eqs. (3.12) and (3.15), it is observed that the $tr(\mathcal{O}^T \mathcal{O})$ in goal-aware robocentric case is greater than that of robocentric mapping. Thus, it can be concluded that

mapping the LMs with respect to the goal provides more information about the system when the system is unobservable, i.e., there is no measurement available from the goal.

3.5 Summary

This Chapter explains the concept of Lie derivatives and provides a detailed 2D observability analysis of the proposed under three different conditions. It is observed that a system containing only the goal is observable as long as the UAV receives measurements from it. Adding any number of landmarks to the system maintains the system's observability as long as the sufficient conditions are met. Finally, a case is explored where the UAV only receives measurements from the LMs but not the goal. It is found that the system becomes unobservable in this case. All these conclusions are validated using simulation results.

Chapter 4

System Overview

The following Chapter outlines the hardware and software components which are used to perform the experiments for our work. Figure. 4.1 shows the system overview, where the blue and orange blocks represent the hardware and software blocks respectively. The hardware components are present on the robot, along with an onboard computer (OBC) which communicates with the Ground Station (GS) containing the user applications. We'll now go over each of these components in detail.

FIGURE 4.1: System Overview

4.1 Robot platform

We use an AR.Drone 2.0 quadcopter for our experiments, as shown in Figure. 4.2. Its main advantage is its low price and robustness to crashes due to the presence of hulls for both outdoor and indoor applications. This comes at the price of flexibility as neither the

hardware itself nor the software running on the OBC can be easily modified. However, this is a great platform for carrying out proof-of-concept experiments such as ours.

The mechanical structure of AR.Drone 2.0 comprises of four brushless 14.5 Watt motors that are attached to the ends of an X-frame quadcopter [103]. The Radio Frequency (RF) hardware, Inertial Measurement Unit (IMU) and the OBC are mounted between the four arms of the UAV and a charged 1000mAh, 11.1V LiPo battery is fastened to it before carrying out any flight tests. The AR.Drone 2.0 monitors battery voltage and converts this into a battery life percentage (100% denotes full, 0% signifying that it has run out of charge). When the UAV detects a low battery voltage, it sends a warning message to the user and then, automatically lands. If it is not in the air, the UAV does not take off if the battery percentage falls below a certain level.

FIGURE 4.2: AR.Drone 2.0 used for experimental testing of this work

FIGURE 4.3: Direction of rotation of four rotors of a quadrotor UAV

The test platform uses two pairs of identical fixed pitched propellers; two clockwise (CW) and two counterclockwise (CCW), as denoted in Figure. 4.3. It hovers or adjusts its altitude by applying equal thrust to all four rotors. Varying left and right rotor speeds the opposite way yields roll movement whereas varying the front and rear rotor speeds yields pitch movement. Yaw movement in one direction is achieved by applying more thrust to rotors rotating in that direction.

The AR.Drone supports three types of communication services - (i) controlling and configuring the drone is done by sending AT commands which are usually sent 30 times per second, (ii) the AR.Drone sends information about its battery percentage, acceleration, etc. to the client (in this case, the GS) 200 times per second, (iii) a video stream is sent by the AR.Drone.

4.2 Sensors

AR.Drone 2.0 is equipped with multiple sensors, most of which are located below the central hull. This includes a 6 Degree-of-Freedom (DOF), MEMS-based miniaturized IMU consisting of gyroscope, accelerometer and magnetometer. These measurements are used by an existing autopilot for roll, pitch and yaw stabilization. An ultrasound sensor is present below the body of the quadcopter which measures its altitude from the ground and assists in automatic altitude stabilization and vertical speed control.

The test UAV has two cameras - (i) a front 720p camera with 93° lens which records up to 30 frames per second (fps), (ii) a downward-pointing Video Graphics Array (VGA) camera with 64° lens which records up to 60 fps. The user can toggle between the cameras based on the application it is being used for. The AR.Drone automatically encodes and streams the video to the host device (mobile app or the GS). Image frames from this stream are used as input to the tracker node described in Section 4.4.1.

4.3 Computer

GS in Figure. 4.1 refers to a Linux laptop running Robot Operating System (ROS). We use a Lenovo ThinkPad with a Quad Core 2.5 Gigahertz (GHz) Intel Core i5 processor as our Ground Station. The GS and OBC communicate with each other over a WiFi network created by the AR.Drone. The network name looks like 'ardrone2_xxx' where 'xxx' is a self-allocated free IP address (typically 192.168.1.1). The high-level trajectory planning is performed on the GS, which are sent to the OBC which runs the low-level autopilot in order to generate the motor control commands.

4.4 Robot Operating System (ROS)

ROS is an open-source platform used widely for various robotic applications. It is a collection of software frameworks used for robot software development. It provides

hardware abstraction, low-level device control, package management and a communication framework, with the help of which various nodes can communicate with each other using a publish-subscribe format. A ROS node refers to a Python or C++ code that executes a subset of the entire application. For example, the Tracker node contains all the necessary code required to execute the computer vision applications of our problem. It 'publishes' the measurements in the form of a topic which can, then, be 'subscribed' by the EKF node in order to be used for its update step. A topic is an instantiation of a message that contains the format in which the data will be published. Some message types are defined within the ROS architecture (e.g, IMU message) whereas others need to be defined specifically for our application purposes, such as the 'States' message in Figure. 4.1.

The blue blocks on the Robot side in Figure. 4.1 are developed by various contributors to the ROS platform. The `ardrone-autonomy` package required to start up and communicate with the test UAV is developed and maintained by the Autonomy Lab of Simon Fraser University [104]. The camera driver supporting image acquisition from the onboard monocular camera uses a standard ROS camera interface [105] and publishes message of type 'image transport' [106]. Each block within the GS represent represent a stand-alone ROS node written using the Python programming language. The details of each of these nodes are described below.

4.4.1 Tracker Node

The Tracker node carries out the goal and landmark (LM) selection and tracking algorithms that are described in Section 2.4. It subscribes to the Image topic published by the camera driver at a rate of 30 Hz. The driver creates three topics out of which the Tracker node subscribes to the `/ardrone/front/image_raw` topic since we use the forward-facing camera to detect both goal and LMs in front of the UAV. This node calculates the bearing measurements based on the tracked locations of the goal and LMs and publishes it using the custom ROS message described in Figure. 4.4. It is important to note here that the Tracker node publishes this data every time it receives an image frame.

```
float64 ekf_init
float64 num_LM
float64 alphag_meas
float64 betag_meas
float64[] lm_gamma
float64[] lm_lambda
```

FIGURE 4.4: Custom ROS message for publishing bearing measurements

Here, *ekf_init* is a flag that represents whether or not to initialize the EKF node which subscribes to this message. When the measurements are received for the first time, *ekf_init* is set to true (or 1) and the measurements are used to initialize the EKF. At all other times, this flag is set to false (or 0). *num_LM* refers to the number of landmarks detected in image frame. The variables *alphag_meas* and *betag_meas* denote the goal's azimuth and elevation angle, and *lm_gamma* and *lm_lambda* denote the measurements from the LMs. All the variables are assigned *float64* data type, where '[']' in the LM variables signifies that they're defined as variable-sized arrays since the number of tracked LMs can vary frame-to-frame.

4.4.2 Estimation Node

The EKF node carries out state estimation as described in Section 2.2. It subscribes to two topics asynchronously. It receives the linear velocity and angular rate data from the */ardrone/imu* topic at 100 Hz and performs the Predict Step. It performs the Update Step upon receiving the measurement data from Tracker node every $1/30^{th}$ of a second. The EKF node publishes the estimated states and their respective covariances to the topic */est_states* at 100 Hz. This topic follows the custom message type shown in Figure. 4.5.

```
float64 alphag_est
float64 betag_est
float64 dg_est
float64 alphag_P
float64 betag_P
float64 dg_P
float64[] lm
float64[] lmPhat
```

FIGURE 4.5: Custom ROS message for publishing estimated states and their covariance

Here, *alphag_est*, *betag_est* and *dg_est* represent the estimated azimuth and elevation bearing, and the distance of the UAV w.r.t the goal respectively. *alphag_P*, *betag_P* and *dg_P* denote their respective covariances. Finally, *lm* and *lmPhat* are the estimated landmark states and their covariances.

4.4.3 Controller Node

The Controller node carries out navigation to goal and obstacle avoidance as discussed in Sections 2.3.1 and 2.3.2. Velocity Controller described in Section 2.3.3 is used only in the

simulator and hence, not written into this ROS node. It subscribes to the `/ardrone/imu` node which provides the forward linear velocity, u . It also subscribes to the `/est_states` topic published by the EKF node. It calculates the commanded yaw rate, r_c using Eqs. (2.31) or (2.37) and generates a fixed commanded forward velocity, u_c . These are, then, published on the `/cmd_vel` topic which uses a predefined ROS message type called 'Twist'.

4.5 Test Facility and Setup

The preliminary hardware tests for our problem are carried out in the indoor Motion Capture room jointly shared by the RISC and MOST-AERO Lab at University of Cincinnati's Victory Parkway campus. This room is equipped with 12 infrared cameras mounted on vertical supports that surround a 6×12 m test area. The actual capture volume of the cameras is slightly lesser than this as objects placed at the very corners of the test area are not visible by sufficient number of cameras that can help triangulate its position. The test area is surrounded by nets and its floor is padded in order to minimize the impact of crashes, if any, at the time of experiments.

FIGURE 4.6: Indoor Motion Capture room

The reflective marker template seen on the top of the AR.Drone in Figure. 4.2 is used for localizing it within this environment. This gives the 'true' or actual position and orientation of the UAV in a 3D coordinate frame. In addition to our robot, we also

position a marker template at or on top of the object that will be selected as the goal. A Python node subscribes to these data in order to calculate the 'true' bearing angle of the UAV w.r.t the goal. This is compared with the result of the EKF node to determine the accuracy of our estimation algorithm. It is important to validate an algorithm with actual or 'true' data in order to troubleshoot any theoretical or programmatic errors. Once the results of the EKF is validated, the controller can be run with the estimated states in any environment with or without the Motion Capture system. An important attribute of our test environment, however, is that it should be feature-rich and should have proper illumination. An empty monotone room, for instance, will have no features that can act as LMs. Even if there are features available, they might be very difficult to track because of their similarity in appearance. Improper lighting can cast shadows and create occlusions on the UAV's path which can hinder its capability to successfully navigate to the goal.

4.6 Simulation

We use MATLAB and Simulink to verify the working of our algorithms in a simulation environment before carrying out the actual hardware tests. We do not define a camera model in the simulator which means that the visual tracking of goal and landmarks which is described in Section 2.4 is not implemented in MATLAB. Instead, it is considered that the locations of goal and LMs are magically provided by some tracker function, which are then used to calculate their respective sensor readings. This is justified because real-time object detection and tracking will pose challenges that cannot always be emulated in the simulation environment. Therefore, we directly test these algorithms on hardware.

FIGURE 4.7: Simulation Block Diagram

Figure. 4.7 shows the different working blocks or functions used in the simulator. A 12-state quadcopter motion model replaces the actual UAV in the simulator. It receives

thrust and torque inputs from the autopilot (detailed in Chapter 6 of [91]). The velocity controller developed in Section 2.3.3 calculates the commanded thrust, roll, pitch and yaw rate based on the outputs of the high-level controller. In the actual AR.Drone, the Velocity Controller and Autopilot blocks are hidden from the user and are performed by its onboard computer itself.

Here, h is given by Eq. (2.21) and

$$X_{quad} = [p_n \ p_e \ p_d \ u \ v \ w \ \phi \ \theta \ \psi \ p \ q \ r]^T$$

$$\hat{X} = [\hat{\alpha}_g \ \hat{\beta}_g \ \hat{d}_g \ \hat{\theta}_i \ \hat{\eta}_i \ \hat{d}_i \ \cdots \ \hat{\theta}_n \ \hat{\eta}_n \ \hat{d}_n]^T$$

4.7 Summary

This Chapter presents an overview of the hardware and software components used in this work. It briefly describes the architecture of Robot Operating System (ROS) in order to illustrate the implementation of our system using actual hardware. It also presents information regarding the test environment and the simulator that are used to verify our proposed solution.

Chapter 5

Results

The following Chapter presents and discusses the results based on both simulation and hardware experiments. We present different test cases within the simulation results such as goal leaving Field-of-View (FOV) and obstacle avoidance. For the hardware experiments, we show the results for visual tracking and navigation to goal in an obstacle-free environment. We analyse the consistency of the system in each of these cases. A filter or state estimator is said to be consistent if [54]:

- The state estimation errors are zero-mean, i.e., the filter is unbiased and
- The filter covariance accurately represents the actual estimation error distribution

The estimation error of a particular state is defined as the difference between its 'True' and 'Estimated' values. The 'True' states are based on some ground truth data which is used to validate the accuracy of the 'Estimated' states generated by the EKF. In simulation, the ground truth is calculated using Eqs. (5.1) - (5.6) whereas, in hardware, this data is obtained from the Motion Capture system described in Section 4.5. The following sections explain each of the cases in detail.

5.1 Simulation Results

Figure. 5.1 shows a snapshot of the Simulink block used for the experiments. This is based on the block diagram shown in Figure. 4.7. Each of the user-defined blocks calls a MATLAB function containing the necessary code. Scopes and '.mat' is used to display and save the output of the required functions. This block diagram remains unchanged irrespective of the case being simulated. The simulation parameters defined in Tables 5.1, 5.2 and 5.3 are defined in a separate parameter file which when executed once saves all

the parameters to the workspace, thus, allowing different blocks or functions such as the EKF or the controller to access these parameters.

FIGURE 5.1: Simulink Block Diagram

5.1.1 Case I - Goal never leaves camera's FOV

FIGURE 5.2: Case I - Initial Conditions. The UAV has bearing measurements from the goal and 5 LMs.

The problem definition assumes that the UAV starts its mission by having the goal in its FOV. In an environment devoid of obstacles, the trajectory defined by the controller will always maintain the goal within the UAV's FOV. Figure. 5.2 shows the initial conditions for this case. The current position of the UAV shows its hover altitude at which the user

selects the goal (denoted by the black '*'). The red circles denote the 3D positions of the 5 landmarks (LMs) detected by the UAV's tracker. The black and red dotted lines represent bearing measurement from the goal and LMs respectively. The UAV is only able to obtain measurements from objects within its FOV.

Parameters	Value
Initial UAV Position $(x_{v_i}, y_{v_i}, z_{v_i})$	(0, 0, 1.2) m
Initial UAV Heading, ψ_{v_i}	$\pi/2$ rad
Goal Position (x_g, y_g, z_g)	(2, 7, 2) m
$LM_1, LM_2, LM_3, LM_4, LM_5$	(1, 8, 1) m, (3.5, 9, 3) m, (-2, 8, 1) m, (2.5, 8, 3) m, (0, 6, 2) m
Bearing Measurement Noise $(\sigma_{az}, \sigma_{el})$	$(0.04, 0.04) \times randn(1)$ rad
Noise in (u, r)	(0.4 m/s, 0.01 rad/s)
Process Noise, Q	0.0025
Initial Bearing Uncertainty $(P_{\alpha_g}, P_{\beta_g}, P_{\theta_1}, P_{\eta_1} \dots P_{\theta_n}, P_{\eta_n})$	$(0.0081, 0.0081, 0.0081, 0.0081,$ $\dots 0.0081, 0.0081) rad^2$
Initial Range Uncertainty $(P_{d_g}, P_{d_1}, \dots P_{d_n})$	$(20, 20, \dots 20) m^2$
Number of Prediction steps in EKF	10
Camera's FOV	60° or $[-\pi/6, \pi/6]$
Navigational Constant, N	3
Desired forward velocity, u_d	0.1 m/s

TABLE 5.1: Parameters used for Simulation Case I

Table 5.1 shows the parameters used for this experiment. It should be noted that the positions of the UAV, its goal and LMs do not mean that the UAV has prior knowledge of its environment. It is considered to have been provided by an internal visual tracker, as described in Section 4.6 and hence, substitutes the implementation of a camera model in the simulator. σ_{az} and σ_{el} refer to the standard deviations in azimuth and elevation bearing angles respectively. Here, $(\alpha_g, \gamma_1, \dots, \gamma_n) \in az$ and $(\beta_g, \lambda_1, \dots, \lambda_n) \in el$. The function *randn* multiplies a random noise between 0 and 1 to σ_{az} and σ_{el} during each sensor measurement. This emulates the inherent noise present in actual sensor data. Finally, P_i refers to the initial uncertainty in i^{th} state.

Figure. 5.4 shows the True versus Estimated state plots for α_g , β_g and d_g . The true states are calculated as follows:

$$\alpha_{true} = \tan^{-1}\left(\frac{y_g - y_v}{x_g - x_v}\right) - \psi_v \quad (5.1)$$

$$\beta_{true} = \sin^{-1}\left(\frac{z_g - z_v}{d_g}\right) \quad (5.2)$$

$$d_{gtrue} = \sqrt{(x_g - x_v)^2 + (y_g - y_v)^2 + (z_g - z_v)^2} \quad (5.3)$$

where, x_v , y_v , z_v and ψ_v represent the current 3D position (m) and heading (rad) of the UAV, respectively. It can be seen that the estimated states follow the true states very closely. The tracking is more accurate in α_g and β_g since the bearing angles are directly measured by the UAV's camera. However, as shown in Figure. 5.5, the errors are always bounded and unbiased signifying that the filter is consistent. The Root Mean Squared (RMS) errors in α_g , β_g and d_g were of the order of 0.0102 rad, 0.0083 rad and 1.6278 m respectively. It is observed via multiple simulations that d_g errors in order of 2 m is good enough for our navigation purposes.

FIGURE 5.3: Case I - 2D view of the UAV's trajectory

Figure 5.3 shows a top-view of the UAV's trajectory to the goal. The measurement lines are better illustrated here than in the 3D view. At the end of its mission, the UAV only receives measurements from the goal, LM_2 and LM_4 . It can be seen that as the UAV maneuvers to the goal, due to its limited FOV, the UAV can no longer track some LMs (LM_3 , LM_5 and eventually LM_1). Due to this, the error between their respective true and estimated states increases, and as expected, their $3 - \sigma$ plots diverge. This can be seen in Figures. 5.6 and 5.7 respectively. It is observed that the $3 - \sigma$ plot of d_3 doesn't diverge as rapidly as those of θ_3 and η_3 . This can be accrued to its already high initial uncertainty which improves until 20 s, after which it slowly starts diverging.

FIGURE 5.4: True Vs Estimated Goal States - α_g , β_g and d_g FIGURE 5.5: $3 - \sigma$ Plots - α_g , β_g and d_g

FIGURE 5.6: True Vs Estimated LM3 States - θ_3 , η_3 and d_3 FIGURE 5.7: 3- σ Plots - θ_3 , η_3 and d_3

FIGURE 5.8: True Vs Estimated LM2 States - θ_2 , η_2 and d_2 FIGURE 5.9: 3 - σ Plots - θ_2 , η_2 and d_2

On the other hand, the UAV can see LM_2 until the end of the mission, which is manifested in Figure. 5.8 as the estimates closely track the true states. d_2 has an uncertainty of ± 10 m at the end, which is slightly high for localization applications. However, when this information is coupled with the significantly lower errors in bearing, the UAV is able to localize the LM with reasonable accuracy. In the above cases, the true landmark states are calculated as follows:

$$\theta_{i_{true}} = \tan^{-1}\left(\frac{y_i - y_v}{x_i - x_v}\right) - \psi_v - \alpha_{true} \quad (5.4)$$

$$\eta_{i_{true}} = \sin^{-1}\left(\frac{z_i - z_v}{d_i}\right) - \beta_{true} \quad (5.5)$$

$$d_{i_{true}} = \sqrt{(x_i - x_v)^2 + (y_i - y_v)^2 + (z_i - z_v)^2} \quad (5.6)$$

where, x_i , y_i and z_i denote the 3D position (m) of the i^{th} LM.

Figures. 5.10 and 5.11 show that the actual forward velocity and yaw rate of the UAV follow the commanded inputs. The commanded signals are generated by the controller mentioned in Section 2.3.1 and the actual data is obtained from the UAV's motion model described in Section 2.3.3.

FIGURE 5.10: Case I - Commanded Vs Actual Forward Velocity (m/s)

FIGURE 5.11: Case I - Commanded Vs Actual Yaw Rate (rad/s)

5.1.2 Case II - Goal leaves camera's FOV (Single Obstacle Avoidance)

As described in Section 2.4.2.2, the UAV will receive no measurements from the goal if it is either occluded by an obstacle or if it leaves the FOV while the UAV tries to circumvent an obstacle. The initial conditions for this case is illustrated in Figure 5.12. We consider a single obstacle located (x_o, y_o, z_o) m away from the origin. It is also assumed that the UAV can sense this obstacle at all times, it only has to avoid it depending on entry and exit criteria defined in Section 2.3.2.

FIGURE 5.12: Case II - Initial Conditions

Parameters	Value
Initial UAV Orientation $(x_{v_i}, y_{v_i}, z_{v_i}, \psi_{v_i})$	(0m, -0.5m, 1.2m $\pi/2$ rad)
Goal Position (x_g, y_g, z_g)	(2, 7, 2) m
$LM_1, LM_2, LM_3, LM_4, LM_5$	(1, 8, 1) m, (3.5, 6, 3) m, (-2, 8, 1) m, (3, 7, 3) m, (0, 6, 2) m
Obstacle Position	(1, 4, 1) m
Collision Radius, r_{col}	2 m
Proportional Gain, k_p	0.5

TABLE 5.2: Parameters used for Simulation Case II

Table 5.2 represents the simulation parameters that are used to run Case II. Rest of the parameters are used exactly the same as in Table 5.1. Figure 5.13 refers to an instant of the simulation when the UAV receives no measurement from the goal. Only LM_2 and LM_4 are within the camera's FOV. The robustness of the EKF is reflected in this case as the filter converges even after having lost direct Line-of-Sight to the goal while trying to avoid the obstacle. As expected, the error between true and estimated states

increases when the goal is not visible, as shown in Figure. 5.15. The filter becomes inconsistent since the errors in α_g , β_g and d_g exceed their respective $3 - \sigma$ bounds. However, once the goal is back in the UAV's FOV, the error between true and estimated states reduces, causing the $3 - \sigma$ plots to converge, as shown in Figure. 5.16. This signifies an improvement in the UAV's belief regarding its relative orientation w.r.t the goal. Figure. 5.14 shows a top-view of the UAV's trajectory. It is clear that the control algorithm switches from PN-guidance to Obstacle Avoidance at S_1 and vice versa at S_2 . In the simulator, a collision is considered to have occurred only when the trajectory or the body of the UAV (sans its arms and hull) run into the obstacle. In practical experiments, however, it is a good idea to keep an additional safety distance S_{col} away from the obstacle's collision radius.

FIGURE 5.13: No measurement received from Goal

FIGURE 5.14: Case II - 2D view of the UAV's Trajectory

FIGURE 5.15: True Vs Estimated Goal States - α_g , β_g and d_g FIGURE 5.16: 3 - σ Plots - α_g , β_g and d_g

Commanded versus Actual forward velocity and yaw rate are illustrated in Figures 5.17 and 5.18 respectively. r_d before S_1 is same as depicted in Figure. 5.11, however, this plot is scaled differently to accommodate the high control inputs generated at the time of controller switching.

FIGURE 5.17: Case II - Commanded Vs Actual Forward Velocity (m/s)

FIGURE 5.18: Case II - Commanded Vs Actual Yaw Rate (rad/s)

5.1.3 Case III - Multiple Obstacle Avoidance

Case II is generalized by placing additional obstacles along the UAV's path to the goal. The simulation parameters except the following are all same as in Section 5.1.1:

Parameters	Value
Initial UAV Orientation $(x_{v_i}, y_{v_i}, z_{v_i}, \psi_{v_i})$	(0m, -0.5m, 1m $\pi/2$ rad)
Goal Position (x_g, y_g, z_g)	(3, 8, 2) m
$LM_1, LM_2, LM_3, LM_4, LM_5$	(2, 8, 1) m, (3.5, 6, 1) m, (-2, 8, 1) m, (3, 7, 2) m, (0, 6, 2) m, (3, 5, 1)
Obstacle Position	(0.2, 2, 1) m, (2, 5, 1) m
Collision Radius, r_{col}	2 m
Proportional Gain, k_p	0.5

TABLE 5.3: Parameters used for Simulation Case III

FIGURE 5.19: Case III - 2D view of the UAV's Trajectory

FIGURE 5.20: True Vs Estimated Goal States - α_g , β_g and d_g

The UAV, now, circumvents two separate obstacles, as seen in Figure. 5.19. The $3 - \sigma$ plots, shown in Figure. 5.21, start to diverge around the 45 s mark, when the UAV is within r_{col} meters of the second obstacle. This is because while trying to avoid this

obstacle, the UAV loses sight of the goal, i.e., $\alpha_g \notin [-\pi/6, \pi/6]$. In case of the first obstacle, however, the goal is still maintained in the camera's FOV. Both these cases are illustrated in the True versus Estimated plot of α_g in Figure. 5.20. The first and second crests represent the UAV's azimuth bearing to the goal while navigating around the first and second obstacle respectively. The errors in all the goal's states are bounded within their respective $3 - \sigma$ bounds, signifying that the system is observable. The RMS errors in α_g , β_g and d_g were obtained as 0.0625 rad, 0.0369 rad and 2.5383 m respectively.

FIGURE 5.21: $3 - \sigma$ Plots - α_g , β_g and d_g

FIGURE 5.22: Case III - Commanded Vs Actual Forward Velocity (m/s)

FIGURE 5.23: Case III - Commanded Vs Actual Yaw Rate (rad/s)

Figures. 5.22 and 5.23 represent the Commanded versus Actual plots for the control inputs u_d, r_d respectively. Similar to Case II, the spikes in these plots depict the controller switching from PN guidance to obstacle avoidance. Once $(\alpha_{obs} - \alpha_g) > \alpha_{diff}$, the controller switches back to PN guidance. In both Case II and III, α_{diff} is set as 0.1 radians.

5.2 Hardware Results

Parameters	Value
Measurement Noise (R)	0.25
Bearing Process Noise	0.25 rad
Range Process Noise	1 m
Initial Bearing Uncertainty ($P_{\alpha_g}, P_{\beta_g}, P_{\theta_1}, P_{\eta_1} \cdots P_{\theta_n}, P_{\eta_n}$)	(0.25, 0.25, 0.25, 0.25, $\cdots$ 0.25, 0.25) rad^2
Initial Range Uncertainty ($P_{d_g}, P_{d_1}, \cdots P_{d_n}$)	(9, 9, $\cdots$ 9) m^2
Number of Prediction steps in EKF	10
Front camera's Horizontal FOV	60° or $[-\pi/6, \pi/6]$
Front camera's Vertical FOV	52.2°
Navigational Constant, N	3
Desired forward velocity, u_d	0.5 m/s
Collision Radius, r_{col}	1 m
Proportional Gain, k_p	0.1 m

TABLE 5.4: Parameters used for Hardware Experiments

FIGURE 5.24: Initial position of UAV before take-off. It is currently tracked by 9 infrared cameras.

FIGURE 5.25: Goal position currently being tracked by 3 infrared cameras.

Figures. 5.24 and 5.25 represent the initial positions of the UAV and the goal respectively, within the Motion Capture room as tracked by the infrared cameras. These figures are screen captured on a separate Windows computer that runs the Motion Capture software, called Motive [107]. The initial positions are kept the same for both Sections 5.2.1 and 5.2.2. A ROS node uses this position data to generate the 'true' bearing and range information of the vehicle with respect to the goal. This is, then, used to evaluate the accuracy of the state estimates generated by the online EKF. The 3D position information is also used to generate the trajectory plots of the vehicle - first, in an obstacle-free environment, followed by an environment with a single obstacle along the UAV's path. In this Thesis, the controller algorithm is validated using the 'true' states. All data generated during experiments are logged using 'rosbag' files which are extracted into

'csv' formats and plotted using MATLAB. The live feed from UAV's onboard camera is recorded via screen capture on the Ground Station (GS). Table 5.4 represents the parameters which are used for the hardware experiments. It should be noted that some of these parameters such as r_{col} and k_p are only used in Section 5.2.2.

5.2.1 Case I: No Obstacle

Figure. 5.26 represents the UAV's camera view at four different time instances during the mission. It can be seen that as the UAV approaches the goal, its template size gradually increases in order to accommodate the increasing size of the object. This is explained in Section 2.4.2.1. Figure. 5.26d shows the template view when the UAV has reached within a certain stopping distance of the goal. The initial view of the goal template when it is first selected by the user is shown in Figure. 5.27.

FIGURE 5.26: Goal Tracking with No Obstacle

Figure. 5.28 represents the UAV's 3D trajectory plot in an obstacle-free environment. The UAV is taken off manually, switched to autonomous mode after selecting the goal. It then follows the trajectory generated by PN-guidance law (Eq. (2.31)) at a fixed forward velocity, u_d . The UAV stops at a d_{stop} of 1 m away from the goal, as illustrated in Figure. 5.29. It should be noted that the stopping boundary represents a circular region of radius 1 m but appears to be elliptical as the X and Y axis are scaled differently.

FIGURE 5.27: Template as selected at the beginning of the mission

FIGURE 5.28: No Obstacle - 3D Trajectory Plot

FIGURE 5.29: No Obstacle - 2D Trajectory Plot

Figure. 5.30 represents the True versus Estimated goal states. The RMS errors in α_g , β_g and d_g were obtained as 0.1640 rad, 0.0136 rad and 0.5635 rad respectively. Figure. 5.32 shows the True versus Estimated plots for the three states of LM_1 . The RMS errors in θ_1 , η_1 and d_1 were obtained as 0.2742 rad, 0.0390 rad and 1.7332 m. The $3 - \sigma$ plots for the goal and LM_1 states are all bounded as shown in Figures. 5.31 and 5.33. This signifies that the state estimator is consistent in actual hardware implementation as well.

FIGURE 5.30: True versus Estimated Plots (No Obstacle) - α_g , β_g , d_g

FIGURE 5.31: $3 - \sigma$ Plots (No Obstacle) - α_g , β_g , d_g

FIGURE 5.32: True versus Estimated Plots (No Obstacle) - θ_1 , η_1 , d_1 FIGURE 5.33: 3 - σ Plots (No Obstacle) - θ_1 , η_1 , d_1

Figure. 5.34 shows the actual range, d_g converging onto desired range which is the same as d_{stop} . Figures. 5.35 and 5.36 depict the Commanded versus Actual forward velocity and yaw rate respectively. Figure. 5.35 shows that the UAV's commanded forward velocity can basically be represented as a binary plot of zero and u_{dmax} velocity. u_d is zero between 0-5 seconds and again after 16 seconds. The first zone represents the UAV's take-off when it only has a velocity along its k^b axis. In the trajectory following zone, the UAV is commanded to fly at a fixed forward velocity of 0.5 m/s. u_d again becomes zero when d_{stop} has been reached and the UAV hovers in front of the goal. It should be noted that the estimation plots in both Sections 5.2.1 and 5.2.2 reflect only the autonomous trajectory following zone, they do not show the manual take-off, hover or landing zone. As a result, the range of time axis in these plots is less than that of the trajectory or Commanded versus Actual plots.

FIGURE 5.34: No Obstacle - Range Convergence Plot

FIGURE 5.35: No Obstacle - Commanded vs Actual Forward Velocity (m/s)

FIGURE 5.36: No Obstacle - Commanded vs Actual Yaw Rate (rad/s)

5.2.2 Case II: Single Obstacle

Keeping the same initial conditions for the UAV and the goal, a virtual obstacle is placed along the UAV's path. In Figure. 5.37a, the orange object denotes the obstacle and for the preliminary tests, it is considered that the UAV knows its location beforehand. The obstacle can also be detected similar to the goal using a template matching technique. This thesis, however, focuses on actively avoiding this obstacle. Figure. 5.37 shows the goal template as tracked by the UAV's camera during different instances of the mission. At $t = 8\text{sec}$, the UAV loses sight of the goal object (a blue chair in this case), however, it is able to efficiently retrack it once the goal reappears in its FOV.

FIGURE 5.37: Goal Tracking during Single Obstacle Avoidance

FIGURE 5.38: Template as selected at the beginning of the mission

The initial view of the template is shown in Figure. 5.38. The obstacle is represented by a red diamond in Figures. 5.39 and 5.40. It can be seen that the UAV efficiently avoids the obstacle and stops at a distance of 1 m away from the goal. The True vs Estimated goal and LM_1 states are represented in Figures. 5.42 and. 5.43. The $3 - \sigma$ plots for all 3 goal and landmark states diverge at the beginning when the UAV is trying to avoid the obstacle, followed by their respective bounds ultimately converging, as shown in Figures. 5.31 and 5.44. This shows that the system is able to estimate the UAV's relative orientation w.r.t these objects with fair accuracy even after receiving no measurements from the goal for a certain period of time. The RMS errors in estimating the goal states α_g , β_g and d_g were calculated to be 0.3240 rad, 0.0383 rad, 0.6341 m respectively. θ_1 , η_1 and d_1 have RMS errors of 0.8886 rad, 0.1260 rad, 0.8107 m respectively.

FIGURE 5.39: Single Obstacle - 3D Trajectory Plot

FIGURE 5.40: Single Obstacle - 2D Trajectory Plot

FIGURE 5.41: True versus Estimated Plots (Single Obstacle) - α_g , β_g , d_g FIGURE 5.42: $3 - \sigma$ Plots (Single Obstacle) - α_g , β_g , d_g FIGURE 5.43: True versus Estimated Plots (Single Obstacle) - θ_1 , η_1 , d_1

FIGURE 5.44: $3-\sigma$ Plots (Single Obstacle) - θ_1 , η_1 , d_1

FIGURE 5.45: Single Obstacle - Range Convergence Plot

FIGURE 5.46: Single Obstacle - Commanded vs Actual Forward Velocity (m/s)

FIGURE 5.47: Single Obstacle - Commanded vs Actual Yaw Rate (rad/s)

Similar to Figure. 5.34, d_g converges to d_{stop} in Figure. 5.45, however, the convergence takes longer in the second case as the UAV maneuvers away from its direct path to goal in order to avoid the obstacle. The UAV's response to control commands is observed to be delayed in comparison to Case I and d_g becomes equal to d_{stop} when the actual forward velocity reaches zero, as can be seen from Figures. 5.45 and 5.46. This delay can also be observed in the Commanded versus Actual yaw rate plot in Figure. 5.47 and can be attributed to the controller tuning. A higher gain should result in faster convergence.

5.3 Summary

This Chapter highlights the problem setup for simulation experiments and provides results for navigation under three different scenarios. The results provided in this Chapter are found to be consistent with the observability analysis in Chapter 3. Based on the experimental setup and software architecture presented in Chapter 4, this Chapter also provides the results from hardware experiments performed in an indoor environment.

Chapter 6

Conclusions and Future Work

This Thesis describes a solution for autonomous navigation of a UAV in a previously unknown and GPS-denied environment in the presence of obstacle(s). In the absence of global data, the problem is formulated using a robot-centered reference frame such that the UAV can use its relative state information w.r.t objects in the environment to fulfill its navigation objectives. The orientation of the landmarks are referenced w.r.t the goal such that it is maintained as the anchor of the map. The relative states are estimated using an Extended Kalman Filter based on a bearing-only measurement model. This mapping technique is shown to provide more information about the system's states than the general robocentric framework. A detailed 2D observability analysis of the system is performed to find the sufficient conditions for the system's observability.

The states of the system are modeled using a 3D coordinate frame, thus, incorporating process noises due to any sudden motion of the UAV. However, to reduce the complexity of state modeling, it is considered that the UAV flies at a constant fixed altitude, i.e, its motion can be defined by controlling its forward velocity and yaw rate. Based on this, a combined PN-guidance and obstacle avoidance algorithm is defined and implemented to help the UAV navigate to the goal while avoiding obstacles along its path. Results presented show the ability of the filter to efficiently estimate the system's states even when there are no measurements available from the goal for a certain period of time. In simulation, a velocity controller is developed to map the high-level control commands to inputs that can be interpreted by an autopilot.

A visual tracking algorithm was developed to track the position of a user-defined goal and other extracted features from the environment. The tracker does not depend on any particular color or intensity of the goal and can track the change in its size due to

increasing proximity of the UAV. The landmarks are tracked using a simple heuristics-based correspondence between consecutive image frames. The visual tracking algorithm is validated using actual hardware.

An obvious extension of this work would be its implementation in outdoor environments solely based on the estimated data. Such environments may be subject to wind disturbances and/or static and dynamic obstacles in the vicinity of the UAV. While dynamic obstacle avoidance is a whole different area of research that might require significant additional computation power, an active static obstacle detection strategy can be implemented using the existing monocular camera onboard. This would require tracking the change in size of obstacles (or landmarks that may act as one) in the same manner as that of goal. When the dimensions exceeds a certain fixed value, the obstacle avoidance technique can be activated. In general, however, single-camera obstacle detection is often challenging due to the absence of optical flow or motion parallax. This can be addressed by incorporating a RGB-D (Red Green Blue Depth) camera or LiDAR sensor which directly provide depth measurements from the environment.

This work can be further extended to incorporate traversal through multiple waypoints or goals. This essentially means that the user would select a new goal after the UAV has reached within a particular stopping distance of the previously selected goal. The current position of the UAV can be thought of as the new origin of the map and the new goal and landmarks will all be mapped w.r.t it. This also means that the size of the state vector will be significantly larger and as such, objects that are not tracked for a certain number of frames need to be removed from it. An association technique that assigns a unique ID to each goal will help distinguish between their respective measurements.

Finally, the system can be modeled so as to incorporate the UAV's body frame velocities v and w along $\hat{j}^b$ and $\hat{k}^b$ axis respectively. This will emulate the complex 3D motions of an aerial vehicle. A robust controller utilizing the 3D velocities and turn rates of the UAV can also be explored in the future.

Bibliography

- [1] Mark W Spong and Mathukumalli Vidyasagar. *Robot dynamics and control*. John Wiley & Sons, 2008.
- [2] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics*. Springer, 2016.
- [3] Kimon P Valavanis. *Advances in unmanned aerial vehicles: state of the art and the road to autonomy*, volume 33. Springer Science & Business Media, 2008.
- [4] Laurence R Newcome. *Unmanned aviation: a brief history of unmanned aerial vehicles*. American Institute of Aeronautics and Astronautics, 2004.
- [5] Luke Dormehl. The history of drones in 10 milestones. <https://www.digitaltrends.com/cool-tech/history-of-drones/>. Last Accessed: 2019-01-23.
- [6] Axel Bürkle. Collaborating miniature drones for surveillance and reconnaissance. In *Unmanned/Unattended Sensors and Sensor Networks VI*, volume 7480, page 74800H. International Society for Optics and Photonics, 2009.
- [7] Randal Beard, Derek Kingston, Timothy W McLain, and Derek Nelson. Decentralized cooperative aerial surveillance using fixed-wing miniature uavs. 2006.
- [8] Derek Kingston, Randal W Beard, and Ryan S Holt. Decentralized perimeter surveillance using a team of uavs. *IEEE Transactions on Robotics*, 24(6):1394–1404, 2008.
- [9] Morgan Quigley, Michael A Goodrich, Stephen Griffiths, Andrew Eldredge, and Randal W Beard. Target acquisition, localization, and surveillance using a fixed-wing mini-uav and gimbaled camera. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE international Conference on*, pages 2600–2605. IEEE, 2005.

- [10] Thomas Kopfstedt, Masakazu Mukai, Masayuki Fujita, and Christoph Ament. Control of formations of uavs for surveillance and reconnaissance missions. *IFAC Proceedings Volumes*, 41(2):5161–5166, 2008.
- [11] Edward Teets, Jr, Casey Donohue, Ken Underwood, and Jeffrey Bauer. Atmospheric considerations for uninhabited aerial vehicle (uav) flight test planning. In *36th AIAA Aerospace Sciences Meeting and Exhibit*, page 748, 1998.
- [12] Lawrence C Trost. *Unmanned air vehicles (UAVs) for cooperative monitoring*. Sandia National Laboratories, 2000.
- [13] Wenqian Zang, Jiayuan Lin, Yangchun Wang, and Heping Tao. Investigating small-scale water pollution with uav remote sensing technology. In *World Automation Congress (WAC), 2012*, pages 1–4. IEEE, 2012.
- [14] Jacy Montenegro M Neto, Ronan A da Paixão, Luiz Renault L Rodrigues, Erick M Moreira, João Carlos José dos Santos, and Paulo Fernando F Rosa. A surveillance task for a uav in a natural disaster scenario. In *2012 IEEE International Symposium on Industrial Electronics*, pages 1516–1522. IEEE, 2012.
- [15] Kostas Alexis, George Nikolakopoulos, Anthony Tzes, and Leonidas Dritsas. Coordination of helicopter uavs for aerial forest-fire surveillance. In *Applications of intelligent control to engineering systems*, pages 169–193. Springer, 2009.
- [16] Stuart M Adams and Carol J Friedland. A survey of unmanned aerial vehicle (uav) usage for imagery collection in disaster research and management. In *9th International Workshop on Remote Sensing for Disaster Response*, page 8, 2011.
- [17] Anuj Puri. A survey of unmanned aerial vehicles (uav) for traffic surveillance. *Department of computer science and engineering, University of South Florida*, pages 1–29, 2005.
- [18] Suman Srinivasan, Haniph Latchman, John Shea, Tan Wong, and Janice McNair. Airborne traffic surveillance systems: video surveillance of highway traffic. In *Proceedings of the ACM 2nd international workshop on Video surveillance & sensor networks*, pages 131–135. ACM, 2004.
- [19] Youngjib Ham, Kevin K Han, Jacob J Lin, and Mani Golparvar-Fard. Visual monitoring of civil infrastructure systems via camera-equipped unmanned aerial vehicles (uavs): a review of related works. *Visualization in Engineering*, 4(1):1, 2016.
- [20] Dieter Hausamann, Werner Zirinig, Gunter Schreier, and Peter Strobl. Monitoring of gas pipelines—a civil uav application. *Aircraft Engineering and Aerospace Technology*, 77(5):352–360, 2005.

- [21] S Sankarasrinivasan, E Balasubramanian, K Karthik, U Chandrasekar, and Rishi Gupta. Health monitoring of civil structures with integrated uav and image processing system. *Procedia Computer Science*, 54:508–515, 2015.
- [22] Najib Metni and Tarek Hamel. A uav for bridge inspection: Visual servoing control law with orientation limits. *Automation in construction*, 17(1):3–10, 2007.
- [23] Norman Hallermann and Guido Morgenthal. Visual inspection strategies for large bridges using unmanned aerial vehicles (uav). In *Proc. of 7th IABMAS, International Conference on Bridge Maintenance, Safety and Management*, pages 661–667, 2014.
- [24] Alan Oxley. *Uncertainties in GPS Positioning: A Mathematical Discourse*. Academic Press, 2017.
- [25] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.
- [26] Randall C Smith and Peter Cheeseman. On the representation and estimation of spatial uncertainty. *The international journal of Robotics Research*, 5(4):56–68, 1986.
- [27] Tim Bailey and Hugh Durrant-Whyte. Simultaneous localization and mapping (slam): Part ii. *IEEE Robotics & Automation Magazine*, 13(3):108–117, 2006.
- [28] Jung-Suk Oh and Kwee-Bo Sim. Symmetrical model based slam [m-slam] for a quick map-search. In *Control Automation and Systems (ICCAS), 2010 International Conference on*, pages 1993–1996. IEEE, 2010.
- [29] Kuen-Han Lin and Chieh-Chih Wang. Stereo-based simultaneous localization, mapping and moving object tracking. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 3975–3980. IEEE, 2010.
- [30] Abraham Bachrach, Samuel Prentice, Ruijie He, and Nicholas Roy. Range-robust autonomous navigation in gps-denied environments. *Journal of Field Robotics*, 28(5):644–666, 2011.
- [31] Paul Newman, John Leonard, Juan D Tardós, and José Neira. Explore and return: Experimental validation of real-time concurrent mapping and localization. In *Robotics and Automation, 2002. Proceedings. ICRA’02. IEEE International Conference on*, volume 2, pages 1802–1809. IEEE, 2002.
- [32] Guoquan P Huang, Anastasios I Mourikis, and Stergios I Roumeliotis. On the complexity and consistency of ukf-based slam. In *Robotics and Automation, 2009. ICRA’09. IEEE International Conference on*, pages 4401–4408. IEEE, 2009.

- [33] Hongjian Wang, Guixia Fu, Juan Li, Zheping Yan, and Xinqian Bian. An adaptive ukf based slam method for unmanned underwater vehicle. *Mathematical Problems in Engineering*, 2013, 2013.
- [34] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In *Proceedings of the 2005 IEEE international conference on robotics and automation*, pages 2432–2437. IEEE, 2005.
- [35] Robert Sim, Pantelis Elinas, Matt Griffin, James J Little, et al. Vision-based slam using the rao-blackwellised particle filter. In *IJCAI Workshop on Reasoning with Uncertainty in Robotics*, volume 14, pages 9–16, 2005.
- [36] Feng Lu and Evangelos Milios. Globally consistent range scan alignment for environment mapping. *Autonomous robots*, 4(4):333–349, 1997.
- [37] Jens-Steffen Gutmann and Bernhard Nebel. Navigation mobiler roboter mit laser-scans. In *Autonome Mobile Systeme 1997*, pages 36–47. Springer, 1997.
- [38] Heung-Yeung Shum, Qifa Ke, and Zhengyou Zhang. Efficient bundle adjustment with virtual key frames: A hierarchical approach to multi-frame structure from motion. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2, pages 538–543. IEEE, 1999.
- [39] Philippe Moutarlier and Raja Chatila. An experimental system for incremental environment modelling by an autonomous mobile robot. In *Experimental Robotics I*, pages 327–346. Springer, 1990.
- [40] John J Leonard and Hugh F Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on robotics and Automation*, 7(3):376–382, 1991.
- [41] MWM Gamini Dissanayake, Paul Newman, Steve Clark, Hugh F Durrant-Whyte, and Michael Csorba. A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on robotics and automation*, 17(3):229–241, 2001.
- [42] Dirk Hähnel, Sebastian Thrun, Ben Wegbreit, and Wolfram Burgard. Towards lazy data association in slam. In *Robotics Research. The Eleventh International Symposium*, pages 421–431. Springer, 2005.
- [43] Michael Bosse and Robert Zlot. Map matching and data association for large-scale two-dimensional laser scan-based slam. *The International Journal of Robotics Research*, 27(6):667–691, 2008.

- [44] Jose Guivant, Eduardo Nebot, and Stephan Baiker. Localization and map building using laser range sensors in outdoor applications. *Journal of robotic systems*, 17(10):565–583, 2000.
- [45] Jose E Guivant and Eduardo Mario Nebot. Optimization of the simultaneous localization and map-building algorithm for real-time implementation. *IEEE transactions on robotics and automation*, 17(3):242–257, 2001.
- [46] John J Leonard and Hans Jacob S Feder. A computationally efficient method for large-scale concurrent mapping and localization. In *Robotics Research*, pages 169–176. Springer, 2000.
- [47] Stefan B Williams, Gamini Dissanayake, and Hugh Durrant-Whyte. An efficient approach to the simultaneous localisation and mapping problem. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 1, pages 406–411. IEEE, 2002.
- [48] Thomas Lemaire, Cyrille Berger, Il-Kyun Jung, and Simon Lacroix. Vision-based slam: Stereo and monocular approaches. *International Journal of Computer Vision*, 74(3):343–364, 2007.
- [49] Guoquan P Huang, Anastasios I Mourikis, and Stergios I Roumeliotis. Analysis and improvement of the consistency of extended kalman filter based slam. In *2008 IEEE International Conference on Robotics and Automation*, pages 473–479. IEEE, 2008.
- [50] Kwang Wee Lee, W Sardha Wijesoma, and Ibanez Guzman Javier. On the observability and observability analysis of slam. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 3569–3574. IEEE, 2006.
- [51] Vincent Aidala and Sherry Hammel. Utilization of modified polar coordinates for bearings-only tracking. *IEEE Transactions on Automatic Control*, 28(3):283–294, 1983.
- [52] José A Castellanos, Ruben Martinez-Cantin, Juan D Tardós, and José Neira. Robocentric map joining: Improving the consistency of ekf-slam. *Robotics and autonomous systems*, 55(1):21–29, 2007.
- [53] Anders Boberg, Adrian N Bishop, and Patric Jensfelt. Robocentric mapping and localization in modified spherical coordinates with bearing measurements. In *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2009 5th International Conference on*, pages 139–144. IEEE, 2009.

- [54] Jeffrey L Ferrin. *Autonomous Goal-Based Mapping and Navigation Using a Ground Robot*. PhD thesis, Brigham Young University, 2016.
- [55] Erann Gat, Rajiv Desai, Robert Ivlev, John Loch, and David P Miller. Behavior control for robotic exploration of planetary surfaces. *IEEE Transactions on Robotics and Automation*, 10(4):490–503, 1994.
- [56] Reid Simmons, Eric Krotkov, Lonnie Chrisman, Fabio Cozman, Richard Goodwin, Martial Hebert, Lalitesh Katragadda, Sven Koenig, Gita Krishnaswamy, Yoshikazu Shinoda, et al. Experience with rover navigation for lunar-like terrains. In *Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, volume 1, pages 441–446. IEEE, 1995.
- [57] James S Jennings, Greg Whelan, and William F Evans. Cooperative search and rescue with a team of mobile robots. In *Advanced Robotics, 1997. ICAR'97. Proceedings., 8th International Conference on*, pages 193–200. IEEE, 1997.
- [58] Clayton Kunz, Chris Murphy, Richard Camilli, Hanumant Singh, John Bailey, Ryan M Eustice, Michael Jakuba, Koichi Nakamura, Chris Roman, Taichi Sato, et al. Deep sea underwater robotic exploration in the ice-covered arctic ocean with auvs. 2008.
- [59] Sebastian Thrun, Scott Thayer, William Whittaker, Christopher Baker, Wolfram Burgard, David Ferguson, Dirk Hahnel, D Montemerlo, Aaron Morris, Zachary Omohundro, et al. Autonomous exploration and mapping of abandoned mines. *IEEE Robotics & Automation Magazine*, 11(4):79–91, 2004.
- [60] Christopher Baker, Aaron Morris, David Ferguson, Scott Thayer, Chuck Whittaker, Zachary Omohundro, Carlos Reverte, William Whittaker, D Hahnel, and Sebastian Thrun. A campaign in autonomous mine mapping. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 2. IEEE, 2004.
- [61] Maxim A Batalin, Gaurav S Sukhatme, and Myron Hattig. Mobile robot navigation using a sensor network. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 1, pages 636–641. IEEE, 2004.
- [62] Yifeng Huang and Kamal Gupta. Rrt-slam for motion planning with motion and map uncertainty for robot exploration. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 1077–1082. IEEE, 2008.

- [63] Adam Bry and Nicholas Roy. Rapidly-exploring random belief trees for motion planning under uncertainty. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 723–730. IEEE, 2011.
- [64] Seyed Abbas Sadat, Kyle Chutskoff, Damir Jungic, Jens Wawerla, and Richard Vaughan. Feature-rich path planning for robust navigation of mavs with mono-slam. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 3870–3875. IEEE, 2014.
- [65] Cindy Leung, Shoudong Huang, and Gamini Dissanayake. Active slam using model predictive control and attractor based exploration. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 5026–5031. IEEE, 2006.
- [66] Paul Zarchan. *Tactical and strategic missile guidance*. American Institute of Aeronautics and Astronautics, Inc., 2012.
- [67] Takeshi Yamasaki, Hirotoshi Sakaida, Keisuke Enomoto, Hiroyuki Takano, and Yoriaki Baba. Robust trajectory-tracking method for uav guidance using proportional navigation. In *Control, Automation and Systems, 2007. ICCAS'07. International Conference on*, pages 1404–1409. IEEE, 2007.
- [68] Su-Cheol Han and Hyochoong Bang. Proportional navigation-based optimal collision avoidance for uavs. In *2nd International Conference on Autonomous Robots and Agents*, pages 13–15, 2004.
- [69] Su-Cheol Han, Hyochoong Bang, and Chang-Sun Yoo. Proportional navigation-based collision avoidance for uavs. *International Journal of Control, Automation and Systems*, 7(4):553–565, 2009.
- [70] Jeffery Saunders, Randal Beard, and Jeffrey Byrne. Vision-based reactive multiple obstacle avoidance for micro air vehicles. In *American Control Conference, 2009. ACC'09.*, pages 5253–5258. IEEE, 2009.
- [71] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *Acm computing surveys (CSUR)*, 38(4):13, 2006.
- [72] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.
- [73] Jianbo Shi and Carlo Tomasi. Good features to track. Technical report, Cornell University, 1993.
- [74] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

- [75] Cor J Veenman, Marcel JT Reinders, and Eric Backer. Resolving motion correspondence for densely moving points. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (1):54–72, 2001.
- [76] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European conference on computer vision*, pages 430–443. Springer, 2006.
- [77] Mustafa Ozuysal, Michael Calonder, Vincent Lepetit, and Pascal Fua. Fast key-point recognition using random ferns. *IEEE transactions on pattern analysis and machine intelligence*, 32(3):448–461, 2010.
- [78] Joseph C Harsanyi and C-I Chang. Hyperspectral image classification and dimensionality reduction: An orthogonal subspace projection approach. *IEEE Transactions on geoscience and remote sensing*, 32(4):779–785, 1994.
- [79] Hua-Liang Wei and Stephen A Billings. Feature subset selection and ranking for data dimensionality reduction. *IEEE transactions on pattern analysis and machine intelligence*, 29(1), 2007.
- [80] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *European conference on computer vision*, pages 778–792. Springer, 2010.
- [81] Jeffrey S Mcveigh and Matthew E Frazer. Mechanism for tracking colored objects in a video sequence, July 6 2004. US Patent 6,760,465.
- [82] Theo Gevers. Robust segmentation and tracking of colored objects in video. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(6):776–781, 2004.
- [83] Parag H Batavia and Sanjiv Singh. Obstacle detection using adaptive color segmentation and color stereo homography. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 1, pages 705–710. IEEE, 2001.
- [84] Yigithan Dedeoglu. Moving object detection, tracking and classification for smart video surveillance. *Master’s Thesis, Bilkent University, Ankara, pp2*, 2004.
- [85] Dorin Comaniciu and Peter Meer. Mean shift analysis and applications. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1197–1203. IEEE, 1999.
- [86] Haim Schweitzer, James Wesley Bell, and Feng Wu. Very fast template matching. In *European Conference on Computer Vision*, pages 358–372. Springer, 2002.

- [87] Feng Tang and Hai Tao. Fast multi-scale template matching using binary features. In *Applications of Computer Vision, 2007. WACV'07. IEEE Workshop on*, pages 36–36. IEEE, 2007.
- [88] Arif Mahmood and Sohaib Khan. Correlation-coefficient-based fast template matching through partial elimination. *IEEE Transactions on image processing*, 21(4):2099, 2012.
- [89] Xi Li, Weiming Hu, Chunhua Shen, Zhongfei Zhang, Anthony Dick, and Anton Van Den Hengel. A survey of appearance models in visual object tracking. *ACM transactions on Intelligent Systems and Technology (TIST)*, 4(4):58, 2013.
- [90] Thomas R Kane and David A Levinson. *Dynamics, theory and applications*. McGraw Hill, 1985.
- [91] Randal W Beard and Timothy W McLain. *Small unmanned aircraft: Theory and practice*. Princeton university press, 2012.
- [92] SN Ghawghawe and D Ghose. Pure proportional navigation against time-varying target manoeuvres. *IEEE Transactions on Aerospace and Electronic Systems*, 32(4):1336–1347, 1996.
- [93] Randal Beard. Quadrotor dynamics and control rev 0.1. 2008.
- [94] Kai Briechle and Uwe D Hanebeck. Template matching using fast normalized cross correlation. In *Optical Pattern Recognition XII*, volume 4387, pages 95–103. International Society for Optics and Photonics, 2001.
- [95] Opencv: Template matching. https://docs.opencv.org/trunk/d4/dc6/tutorial_py_template_matching.html. Last Accessed: 2018-03-20.
- [96] Edward R Dougherty. An introduction to morphological image processing (tutorial texts in optical engineering. DC O'Shea, SPIE Optical Engineering Press, Bellingham, WA, USA, 1992.
- [97] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.
- [98] Adrian Rosebrock. Simple object tracking with opencv. <https://www.pyimagesearch.com/2018/07/23/simple-object-tracking-with-opencv/>. Last Accessed: 2018-12-26.
- [99] J Karl Hedrick and Anouck Girard. Control of nonlinear dynamic systems: Theory and applications. *Controllability and observability of Nonlinear Systems*, 48, 2005.

- [100] Robert Hermann and Arthur Krener. Nonlinear controllability and observability. *IEEE Transactions on automatic control*, 22(5):728–740, 1977.
- [101] Gilbert Strang, Gilbert Strang, Gilbert Strang, and Gilbert Strang. *Introduction to linear algebra*, volume 3. Wellesley-Cambridge Press Wellesley, MA, 1993.
- [102] Junjian Qi, Kai Sun, and Wei Kang. Optimal pmu placement for power system dynamic state estimation by using empirical observability gramian. *IEEE Transactions on power Systems*, 30(4):2041–2054, 2015.
- [103] Stephane Piskorski, Nicolas Brulez, Pierre Eline, and Frederic DHaeyer. Ar. drone developer guide. *Parrot, sdk*, 1, 2012.
- [104] Autonomy Lab. ardrone-autonomy. <https://ardrone-autonomy.readthedocs.io/en/latest/>. Last Accessed: 2018-12-26.
- [105] Jack O’Quin. camera_drivers. http://wiki.ros.org/camera_drivers. Last Accessed: 2018-12-26.
- [106] Patrick Mihelich. image_transport. http://wiki.ros.org/image_transport. Maintained by Jack O’Quin, Vincent Rabaud, Last Accessed: 2018-12-26.
- [107] OptiTrack. Motive:tracker. <https://optitrack.com/products/motive/tracker/>. Last Accessed: 2019-02-13.
- [108] Joao P Hespanha. Lecture notes on lqr/lqg controller design. *Knowledge creation diffusion utilization*, 2005.
- [109] Graham C Goodwin, Stefan F Graebe, Mario E Salgado, et al. *Control system design*. Upper Saddle River, NJ: Prentice Hall,, 2001.
- [110] MathWorks. lqr. <https://www.mathworks.com/help/control/ref/lqr.html>. Last Accessed: 2019-02-17.

Experimental Challenges

The following appendix list some of the issues faced during hardware implementation of the proposed solution. They are highlighted as follows:

- AR.Drone 2.0 generates its own wireless network of 2.4GHz bandwidth. All sensor data from the UAV as well as the control commands from the Ground Station (GS) are transmitted over this network. The Motion Capture System (Mocap) also streams positioning data over a 2.4GHz channel. This was seen to cause significant communication interference or delays as data transmission rates would drop considerably. This issue was more pronounced while reading the real-time image data from AR.Drone. While the images are supposed to stream at 30 fps (frames per second), the rate was seldom observed to be more than 2Hz, resulting in the tracking algorithm receiving no inputs to carry out its online processing. This was solved by disabling the wireless network streaming from the router configuration settings and using a wired network to transmit the Mocap data. Simply using an Ethernet cable without disabling the wireless network didn't solve this issue.
- During its flight, a quadcopter's propellers displace the air around them. Within the indoor Mocap room, this would sometimes result in significant vortices that would displace the UAV from its original hovering position. This would make it difficult to correctly observe and select the goal object from its camera feed or hover a fixed distance away from the goal.
- AR.Drone is a low-cost commercial off-the-shelf (COTS) product that is widely used to carry out proof-of-concept hardware experiments on multicopters. Yet, the biggest disadvantage is the inability to access their internal components. Even though there is rigorous documentation available regarding operating it in different modes, there is no way to exploit the processing prowess of its on-board computer and as such, all software developed for this work run explicitly from the GS. Any communication failure between the UAV and GS would result in complete data loss and possible mission failures. Ability to integrate a single-board computer to

the test vehicle will enable certain mission-critical algorithms to be run on it, thus, reducing the chances of failure.

- The data from the infrared cameras can be visualized using a software called Motive, which runs on a different computer than the GS. The position of the UAV and goal marker templates are used to create 'assets' which are, then, used by this software to continuously track the templates. Long periods of inactivity on this system was seen to ultimately result in no data streaming, thus, providing no 'ground truth' information for validating the results. The assets needed to be reassigned to the marker templates several times during the experiments to ensure uninterrupted streaming of position data.

Linear Quadratic Regulator (LQR)

Linear-Quadratic Regulator (LQR) is a state-feedback controller commonly used to minimize the cost index of operating a dynamic system. It has been covered extensively in literature [91, 108, 109] and hence, this text will only cover the LQR gain calculation.

Given the state space equation

$$\dot{x} = Ax + Bu \quad (1)$$

the LQR cost can be given by:

$$J(x_0) = \min_{u(t), t \geq 0} \int_0^\infty x^T(\tau)Qx(\tau) + u^T(\tau)Ru(\tau)d\tau \quad (2)$$

where, $Q \geq 0$ is a symmetric, positive semi-definite matrix and $R > 0$ is a symmetric, positive definite matrix.

If (A, B) is controllable and $(A, Q^{1/2})$ is observable, then the control input can be written as [91]:

$$u^*(t) = -K_{lqr}x(t) \quad (3)$$

The LQR gain K_{lqr} can be calculated as:

$$K_{lqr} = R^{-1}B^TP \quad (4)$$

where, P is a symmetric positive definite matrix that can be obtained by solving the Algebraic Ricatti Equation

$$PA + A^TP + Q - PBR^{-1}B^TP = 0 \quad (5)$$

Q and R are tuning parameters and can be changed to meet a desired system requirement. Higher weights of Q implies that more emphasis should be put on optimizing the system states whereas a higher R will give more importance to the control inputs and

will try to minimize the errors significantly faster than in a system with higher Q . Q and R matrices can be written as:

$$Q = \begin{bmatrix} q_1 & 0 & \cdots & 0 \\ 0 & q_2 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & q_n \end{bmatrix}, \quad R = \begin{bmatrix} r_1 & 0 & \cdots & 0 \\ 0 & r_2 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & r_m \end{bmatrix}$$

where, n is the number of states and m is the number of inputs. Choosing $q_i \geq 0$ and $r_i > 0$ ensures that Q and R are positive semi-definite and positive definite respectively. A simple starting point for tuning Q and R is given by Bryson's rule [108]:

$$Q_{ii} = \frac{1}{\text{maximum acceptable value of } x_i^2} \quad (6)$$

$$R_{jj} = \frac{1}{\text{maximum acceptable value of } u_j^2} \quad (7)$$

Based on the control performance obtained by these values of Q and R , they might be further tuned to obtain desirable properties of the closed-loop system. In MATLAB, the value of K_{lqr} can be generated using the command: $[K, P, E] = lqr(A, B, Q, R)$ [110].

Declaration of Authorship

I, Srijanee Biswas, declare that this thesis titled, ‘Goal-Aware Robocentric Mapping and Navigation of a Quadrotor Unmanned Aerial Vehicle’ and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: DocuSigned by:
418002BC2903422...

Date: 3/12/2019