Swarm Localization and Control via On-board Sensing and Computation

Thesis by
Fat-Hy Omar Rajab

In Partial Fulfillment of the Requirements

For the Degree of
Masters of Science

King Abdullah University of Science and Technology
Thuwal, Kingdom of Saudi Arabia

July, 2019
The thesis of Fat-Hy Omar Rajab is approved by the examination committee

Committee Chairperson: Prof. Jeff S. Shamma
Committee Members: Prof. Taous-Meriem Laleg-Kirati, Prof. Shehab Ahmad
ABSTRACT

Swarm Localization and Control via On-board Sensing and Computation
Fat-Hy Omar Rajab

Multi-agent robotic system have been proved to be more superior in undertaking functionalities, arduous or even impossible when performed by single agents. The increased efficiency in multi agent systems is achieved by the execution of the task in cooperative manner. But to achieve cooperation in multi agent systems, a good localization system is an important prerequisite. Currently, most of the multi-agent system rely on the use of the GPS to provide global positioning information which suffers great deterioration in performance in indoor applications, and also all to all communication between the agents will be required which is not efficient especially when the number of agents is large. In this regard, a real-time localization scheme is introduced which makes use of the robot’s on-board sensors and computational capabilities to determine the states of other agents in the multi agent system. This algorithm also takes the advantage of the swarming behaviour of the robots in the estimation of the states. This localization algorithm was found to produce more accurate agent state estimates as compared to a similar localization algorithm that does not take into account the swarming behaviour of the agents in simulations and real experiment involving two Unmanned Aerial Vehicles.
ACKNOWLEDGEMENTS

Firstly, I thank God for helping me accomplish this challenging and yet interesting task in good health.

I would like to extend my sincere gratitude to the King Abdullah University of Science and Technology, for giving me a great opportunity to benefit in high quality education and research facilities and giving me a platform to learn from world top-notch scholars and contribute in pushing the limits of human knowledge.

A lot of thanks goes to my advisor, Professor Jeff Shamma whom I benefit a lot from his vast knowledge and wisdom. He has always been patiently forging and sharpening my understanding of different concepts, being my harshest critic and thus stimulating the best performance from me.

I also thank my Masters defence committee members, Professor Shehab Ahmed and Professor Meriem Laleg, for accepting to share with me a slice of their valuable time to look at my work. I have personally learned a lot from different courses offered by them in the university and looking forward for further collaboration with them in my PhD.

My fellow RISC lab members have had a big impact and improved me a lot. I would especially thank Dr. Mohammed Abdelkader, Dr. Samet Guler, and Kuat Telegenov for their great support and contribution to my work. I would like to thank all the other RISC lab members especially the ERL team 2019, It is always a great pleasure working with you.

Finally, I would like to thank my whole family for their unwavering support and encouragement, especially my parents Mr. Omar Rajab and Mrs. Moza Ali, and my siblings, Munira, Khairat, Nadhifat, Sharifa, Khaulat, and Ridhwan, for always being
on my side and including me in their prayers. And most importantly I would like to give a huge appreciation to my love of my life, my wife Rayyan Abdallah and all her family for always being understanding and supportive throughout my work.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Examination Committee Page</td>
<td>2</td>
</tr>
<tr>
<td>Copyright</td>
<td>3</td>
</tr>
<tr>
<td>Abstract</td>
<td>4</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>5</td>
</tr>
<tr>
<td>List of Figures</td>
<td>10</td>
</tr>
<tr>
<td>List of Tables</td>
<td>15</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>16</td>
</tr>
<tr>
<td>1.1 Literature Review</td>
<td>17</td>
</tr>
<tr>
<td>1.2 Objectives and Contributions</td>
<td>20</td>
</tr>
<tr>
<td>1.3 Thesis Outline</td>
<td>21</td>
</tr>
<tr>
<td>2 Modeling of a Quadrotor</td>
<td>22</td>
</tr>
<tr>
<td>2.1 Kinematics of a Quadcopter</td>
<td>23</td>
</tr>
<tr>
<td>2.1.1 Quadcopter Attitude with respect to the Inertial Frame</td>
<td>25</td>
</tr>
<tr>
<td>2.2 Dynamics of a Quadcopter</td>
<td>31</td>
</tr>
<tr>
<td>2.2.1 Forces and Torques acting on the body of a Quadcopter</td>
<td>32</td>
</tr>
<tr>
<td>2.2.2 Gyscopic Moments</td>
<td>34</td>
</tr>
<tr>
<td>2.3 Translation Equations of Motion</td>
<td>37</td>
</tr>
<tr>
<td>2.4 Rotational Equations of Motion</td>
<td>39</td>
</tr>
<tr>
<td>3 Position and Formation Control of Quadcopter</td>
<td>43</td>
</tr>
<tr>
<td>3.1 Position Control of a Quadcopter</td>
<td>43</td>
</tr>
<tr>
<td>3.2 Formation Control of a Quadcopter</td>
<td>47</td>
</tr>
<tr>
<td>4 Swarm Localization</td>
<td>52</td>
</tr>
<tr>
<td>4.1 Introduction to Ultra-Wide Band (UWB) Technology</td>
<td>52</td>
</tr>
<tr>
<td>4.1.1 Characteristics of the UWB technology</td>
<td>55</td>
</tr>
</tbody>
</table>
4.1.2 Comparison with Other Widely Used Ranging Technologies

4.2 Radio Frequency Ranging Methodologies

4.2.1 Signal Strength

4.2.2 Angle of arrival

4.2.3 Time Difference of Flight

4.2.4 Time Of Flight

4.3 Trilateration

4.4 Filtering

4.4.1 Exponential Smoothing

4.4.2 Kalman Filter

4.5 Peer-to-Peer Localization between Agents in a Multi-Agent System

4.5.1 Drift Localization Algorithm

4.5.2 Simulation Results

4.5.3 Swarm Localization Algorithm

4.5.4 Simulation Results

4.6 Comparison Between the Drift and the Swarm Localization using simulation results

4.6.1 Effect of Changing the Error Covariance Matrix

4.6.2 Effect of Changing the Process Noise Covariance Matrix

4.6.3 Effect of Changing the Observation Noise Covariance Matrix

5 Experiments

5.1 Apparatus used for the Experiment

5.1.1 Decawave

5.1.2 Motion Capture System

5.1.3 Joystick

5.1.4 Control Center Computer

5.1.5 On-board Computer

5.1.6 Lidar Lite Range Finder

5.1.7 Airframe

5.2 Experimental Procedure

5.2.1 Calibration

5.3 Programs used for the Experiments

5.4 Results of the Experiments

5.4.1 Effect of Changing the Error Covariance Matrix

5.4.2 Effect of Changing the Process Noise Covariance Matrix
6 Obstacle Avoidance

6.1 Objective of the ERL tournament in February, 2019

6.1.1 Existing Works in Obstacle avoidance

6.1.2 Obstacle Avoidance using the RPLidar

6.2 Autonomous navigation with Obstacle Avoidance

6.2.1 Existing Works in Obstacle avoidance

6.2.2 Obstacle Avoidance using the RPLidar

6.2.3 Outdoor Obstacle Avoidance Algorithm

6.2.4 Indoor Obstacle Avoidance Algorithm

7 Concluding Remarks

7.1 Summary

7.2 Future Research Work

References

Appendices
### LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure Number</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Kinematics of Quadcopter</td>
<td>23</td>
</tr>
<tr>
<td>3.1</td>
<td>Position X control using PI controller in MATLAB</td>
<td>45</td>
</tr>
<tr>
<td>3.2</td>
<td>Position Y control using PI controller in MATLAB</td>
<td>46</td>
</tr>
<tr>
<td>3.3</td>
<td>Position X control using PI controller in ROS and Gazebo</td>
<td>46</td>
</tr>
<tr>
<td>3.4</td>
<td>Position Y control using PI controller in ROS and Gazebo</td>
<td>47</td>
</tr>
<tr>
<td>3.5</td>
<td>Formation control in X using PI controller in MATLAB/Simulink</td>
<td>49</td>
</tr>
<tr>
<td>3.6</td>
<td>Formation control in Y using PI controller in MATLAB/Simulink</td>
<td>50</td>
</tr>
<tr>
<td>3.7</td>
<td>Formation control in X using PI controller in ROS and Gazebo</td>
<td>50</td>
</tr>
<tr>
<td>3.8</td>
<td>Formation control in Y using PI controller in ROS and Gazebo</td>
<td>51</td>
</tr>
<tr>
<td>4.1</td>
<td>Power Density limit Enacted by FCC in UWB systems (From [2])</td>
<td>55</td>
</tr>
<tr>
<td>4.2</td>
<td>Resolution Comparison Between Positioning Technologies (From [3])</td>
<td>56</td>
</tr>
<tr>
<td>4.3</td>
<td>Two Way Ranging Protocol (From [4])</td>
<td>61</td>
</tr>
<tr>
<td>4.4</td>
<td>Symmetric Double-sided Two Way Ranging Protocol (From [4])</td>
<td>62</td>
</tr>
<tr>
<td>4.5</td>
<td>X position estimate using Drift Localization with $\sigma_p = 1, \sigma_q = 0.001^2$ and $\sigma_r = 0.0005^2$ in Simulation</td>
<td>75</td>
</tr>
<tr>
<td>4.6</td>
<td>Y position estimate using Drift Localization with $\sigma_p = 1, \sigma_q = 0.001^2$ and $\sigma_r = 0.0005^2$ in Simulation</td>
<td>75</td>
</tr>
<tr>
<td>4.7</td>
<td>Vx velocity estimate using Drift Localization with $\sigma_p = 1, \sigma_q = 0.001^2$ and $\sigma_r = 0.0005^2$ in Simulation</td>
<td>76</td>
</tr>
<tr>
<td>4.8</td>
<td>Vy velocity estimate using Drift Localization with $\sigma_p = 1, \sigma_q = 0.001^2$ and $\sigma_r = 0.0005^2$ in Simulation</td>
<td>76</td>
</tr>
<tr>
<td>4.9</td>
<td>X position estimate using Swarm Localization with $\sigma_p = 1, \sigma_q = 0.001^2$ and $\sigma_r = 0.05^2$</td>
<td>81</td>
</tr>
<tr>
<td>4.10</td>
<td>Y position estimate using Swarm Localization with $\sigma_p = 1, \sigma_q = 0.001^2$ and $\sigma_r = 0.05^2$ in Simulation</td>
<td>81</td>
</tr>
<tr>
<td>4.11</td>
<td>Vx velocity estimate using Swarm Localization with $\sigma_p = 1, \sigma_q = 0.001^2$ and $\sigma_r = 0.05^2$ in Simulation</td>
<td>82</td>
</tr>
<tr>
<td>4.12</td>
<td>Vy velocity estimate using Swarm Localization with $\sigma_p = 1, \sigma_q = 0.001^2$ and $\sigma_r = 0.05^2$ in Simulation</td>
<td>82</td>
</tr>
</tbody>
</table>
4.13 X position estimate using Drift Localization with $\sigma_p = 10, \sigma_q = 0.001^2$ and $\sigma_r = 0.05^2$ in Simulation

4.14 X position estimate using Swarm Localization with $\sigma_p = 10, \sigma_q = 0.001^2$ and $\sigma_r = 0.0001^2$ in Simulation

4.15 Y position estimate using Drift Localization with $\sigma_p = 10, \sigma_q = 0.001^2$ and $\sigma_r = 0.05^2$ in Simulation

4.16 Y position estimate using Swarm Localization with $\sigma_p = 10, \sigma_q = 0.001^2$ and $\sigma_r = 0.0005^2$ in Simulation

4.17 Vx velocity estimate using Drift Localization with $\sigma_p = 10, \sigma_q = 0.001^2$ and $\sigma_r = 0.05^2$ in Simulation

4.18 Vx velocity estimate using Swarm Localization with $\sigma_p = 10, \sigma_q = 0.0001^2$ and $\sigma_r = 0.0005^2$ in Simulation

4.19 Vy velocity estimate using Drift Localization with $\sigma_p = 10, \sigma_q = 0.001^2$ and $\sigma_r = 0.05^2$ in Simulation

4.20 Vy velocity estimate using Swarm Localization with $\sigma_p = 10, \sigma_q = 0.0001^2$ and $\sigma_r = 0.0005^2$ in Simulation

4.21 X position estimate using Drift Localization with $\sigma_p = 1, \sigma_q = 0.0001^2$ and $\sigma_r = 0.05^2$ in Simulation

4.22 X position estimate using Swarm Localization with $\sigma_p = 1, \sigma_q = 0.0001^2$ and $\sigma_r = 0.0005^2$ in Simulation

4.23 Y position estimate using Drift Localization with $\sigma_p = 1, \sigma_q = 0.0001^2$ and $\sigma_r = 0.05^2$ in Simulation

4.24 Y position estimate using Swarm Localization with $\sigma_p = 1, \sigma_q = 0.0001^2$ and $\sigma_r = 0.0005^2$ in Simulation

4.25 Vx velocity estimate using Drift Localization with $\sigma_p = 1, \sigma_q = 0.0001^2$ and $\sigma_r = 0.05^2$ in Simulation

4.26 Vx velocity estimate using Swarm Localization with $\sigma_p = 1, \sigma_q = 0.0001^2$ and $\sigma_r = 0.0005^2$ in Simulation

4.27 Vy velocity estimate using Drift Localization with $\sigma_p = 1, \sigma_q = 0.0001^2$ and $\sigma_r = 0.0005^2$ in Simulation

4.28 Vy velocity estimate using Swarm Localization with $\sigma_p = 1, \sigma_q = 0.0001^2$ and $\sigma_r = 0.0005^2$ in Simulation
4.31 Y position estimate using Drift Localization with $\sigma_p = 1, \sigma_q = 0.001^2$
and $\sigma_r = 0.0005^2$ in Simulation

95

4.32 Y position estimate using Swarm Localization with $\sigma_p = 1, \sigma_q = 0.001^2$
and $\sigma_r = 0.0005^2$ in Simulation

95

4.33 Vx velocity estimate using Drift Localization with $\sigma_p = 1, \sigma_q = 0.001^2$
and $\sigma_r = 0.0005^2$ in Simulation

96

4.34 Vx velocity estimate using Swarm Localization with $\sigma_p = 1, \sigma_q = 0.001^2$
and $\sigma_r = 0.0005^2$ in Simulation

96

4.35 Vy velocity estimate using Drift Localization with $\sigma_p = 1, \sigma_q = 0.001^2$
and $\sigma_r = 0.0005^2$ in Simulation

97

4.36 Vy velocity estimate using Swarm Localization with $\sigma_p = 1, \sigma_q = 0.001^2$
and $\sigma_r = 0.0005^2$ in Simulation

97

5.1 Anchor Drone used for the Indoor Experimental setup

101

5.2 Tag Drone Used for the Indoor Experimental Setup

102

5.3 Indoor Experimental setup for Swarm Localization Algorithm

104

5.4 Indoor Experimental setup for Drift Localization Algorithm

104

5.5 X position estimate using Drift and Swarm Localization with $\sigma_p = 10, \sigma_q = 0.001^2, \text{ and } \sigma_r = 0.05^2$ in Indoor Experiment

107

5.6 X position estimate using Drift and Swarm Localization with $\sigma_p = 0.1, \sigma_q = 0.001^2, \text{ and } \sigma_r = 0.05^2$ in Indoor Experiment

107

5.7 Y position estimate using Drift and Swarm Localization with $\sigma_p = 10, \sigma_q = 0.001^2, \text{ and } \sigma_r = 0.05^2$ in Indoor Experiment

108

5.8 Y position estimate using Drift and Swarm Localization with $\sigma_p = 0.1, \sigma_q = 0.001^2, \text{ and } \sigma_r = 0.05^2$ in Indoor Experiment

108

5.9 Vx velocity estimate using Drift and Swarm Localization with $\sigma_p = 10, \sigma_q = 0.001^2, \text{ and } \sigma_r = 0.05^2$ in Indoor Experiment

109

5.10 Vx velocity estimate using Drift and Swarm Localization with $\sigma_p = 0.1, \sigma_q = 0.001^2, \text{ and } \sigma_r = 0.05^2$ in Indoor Experiment

109

5.11 Vy velocity estimate using Drift and Swarm Localization with $\sigma_p = 10, \sigma_q = 0.001^2, \text{ and } \sigma_r = 0.05^2$ in Indoor Experiment

110

5.12 Vy velocity estimate using Drift and Swarm Localization with $\sigma_p = 0.1, \sigma_q = 0.001^2, \text{ and } \sigma_r = 0.05^2$ in Indoor Experiment

110

5.13 X position estimate using Drift and Swarm Localization with $\sigma_p = 1, \sigma_q = 0.0001^2, \text{ and } \sigma_r = 0.05^2$ in Indoor Experiment

112
5.14 X position estimate using Drift and Swarm Localization with $\sigma_p = 1, \sigma_q = 0.01^2$ and $\sigma_r = 0.05^2$ in Indoor Experiment  

5.15 X position estimate using Drift and Swarm Localization with $\sigma_p = 1, \sigma_q = 0.1^2$ and $\sigma_r = 0.05^2$ in Indoor Experiment  

5.16 Y position estimate using Drift and Swarm Localization with $\sigma_p = 1, \sigma_q = 0.0001^2$ and $\sigma_r = 0.05^2$ in Indoor Experiment  

5.17 Y position estimate using Drift and Swarm Localization with $\sigma_p = 1, \sigma_q = 0.01^2$ and $\sigma_r = 0.05^2$ in Indoor Experiment  

5.18 Y position estimate using Drift and Swarm Localization with $\sigma_p = 1, \sigma_q = 0.1^2$ and $\sigma_r = 0.05^2$ in Indoor Experiment  

5.19 Vx velocity estimate using Drift and Swarm Localization with $\sigma_p = 1, \sigma_q = 0.0001^2$ and $\sigma_r = 0.05^2$ in Indoor Experiment  

5.20 Vx position estimate using Drift and Swarm Localization with $\sigma_p = 1, \sigma_q = 0.01^2$ and $\sigma_r = 0.05^2$ in Indoor Experiment  

5.21 Vx velocity estimate using Drift and Swarm Localization with $\sigma_p = 1, \sigma_q = 0.1^2$ and $\sigma_r = 0.05^2$ in Indoor Experiment  

5.22 Vy velocity estimate using Drift and Swarm Localization with $\sigma_p = 1, \sigma_q = 0.0001^2$ and $\sigma_r = 0.05^2$ in Indoor Experiment  

5.23 Vy position estimate using Drift and Swarm Localization with $\sigma_p = 1, \sigma_q = 0.01^2$ and $\sigma_r = 0.05^2$ in Indoor Experiment  

5.24 Vy velocity estimate using Drift and Swarm Localization with $\sigma_p = 1, \sigma_q = 0.1^2$ and $\sigma_r = 0.05^2$ in Indoor Experiment  

5.25 X position estimate using Drift and Swarm Localization with $\sigma_p = 1, \sigma_q = 0.001^2$ and $\sigma_r = 0.005^2$ in Indoor Experiment  

5.26 X position estimate using Drift and Swarm Localization with $\sigma_p = 1, \sigma_q = 0.001^2$ and $\sigma_r = 0.0005^2$ in Indoor Experiment  

5.27 Y position estimate using Drift and Swarm Localization with $\sigma_p = 1, \sigma_q = 0.001^2$ and $\sigma_r = 0.05^2$ in Indoor Experiment  

5.28 Y position estimate using Drift and Swarm Localization with $\sigma_p = 1, \sigma_q = 0.001^2$ and $\sigma_r = 0.0005^2$ in Indoor Experiment  

5.29 Vx velocity estimate using Drift and Swarm Localization with $\sigma_p = 1, \sigma_q = 0.001^2$ and $\sigma_r = 0.05^2$ in Indoor Experiment  

5.30 Vx velocity estimate using Drift and Swarm Localization with $\sigma_p = 1, \sigma_q = 0.001^2$ and $\sigma_r = 0.0005^2$ in Indoor Experiment  

5.31 Vy velocity estimate using Drift and Swarm Localization with $\sigma_p = 1, \sigma_q = 0.001^2$ and $\sigma_r = 0.05^2$ in Indoor Experiment
5.32 Vy velocity estimate using Drift and Swarm Localization with $\sigma_p = 1 \cdot 10^{-1}$, $\sigma_q = 0.001^2$ and $\sigma_r = 0.0005^2$ in Indoor Experiment

6.1 Mode of Action of RPLidar (From [5])
6.2 Outdoor Obstacle Avoidance Algorithm
6.3 Outdoor Obstacle Avoidance Algorithm

A.1 Drone simulation
A.2 PI position controller in drone simulation
A.3 Anchor-Tag drone simulation
A.4 PI Formation Controller in the Tag Drone
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Comparison of short range Positioning Technology (From [4])</td>
<td>57</td>
</tr>
<tr>
<td>4.2</td>
<td>RMSE in state estimates using Drift Localization Algorithm as $\sigma_p$ varies in Simulation</td>
<td>83</td>
</tr>
<tr>
<td>4.3</td>
<td>RMSE in state estimates using Swarm Localization Algorithm as $\sigma_p$ varies in Simulation</td>
<td>83</td>
</tr>
<tr>
<td>4.4</td>
<td>RMSE in state estimates using Drift Localization Algorithm as $\sigma_q$ varies in Simulation</td>
<td>88</td>
</tr>
<tr>
<td>4.5</td>
<td>RMSE in state estimates using Swarm Localization Algorithm as $\sigma_q$ varies in Simulation</td>
<td>88</td>
</tr>
<tr>
<td>4.6</td>
<td>RMSE in state estimates using Drift Localization Algorithm as $\sigma_r$ varies in Simulation</td>
<td>93</td>
</tr>
<tr>
<td>4.7</td>
<td>RMSE in state estimates using Swarm Localization Algorithm as $\sigma_r$ varies in Simulation</td>
<td>94</td>
</tr>
<tr>
<td>5.1</td>
<td>RMSE in state estimates using Drift Localization Algorithm as $\sigma_p$ varies in Indoor Experiments</td>
<td>106</td>
</tr>
<tr>
<td>5.2</td>
<td>RMSE in state estimates using Swarm Localization Algorithm as $\sigma_p$ varies in Indoor Experiments</td>
<td>106</td>
</tr>
<tr>
<td>5.3</td>
<td>RMSE in state estimates using Drift Localization Algorithm as $\sigma_q$ varies in Indoor Experiments</td>
<td>111</td>
</tr>
<tr>
<td>5.4</td>
<td>RMSE in state estimates using Swarm Localization Algorithm as $\sigma_q$ varies in Indoor Experiments</td>
<td>111</td>
</tr>
<tr>
<td>5.5</td>
<td>RMSE in state estimates using Drift Localization Algorithm as $\sigma_r$ varies in Indoor Experiments</td>
<td>118</td>
</tr>
<tr>
<td>5.6</td>
<td>RMSE in state estimates using Swarm Localization Algorithm as $\sigma_r$ varies in Indoor Experiments</td>
<td>119</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

The emergence of multi-agent systems in the field of robotics has found many applications in numerous fields including commercial, public services, and military sectors [6]. Multi-agent systems have the potential to demonstrate far more superior efficiency compared to single-agent systems in performing different tasks due to their ability to cooperate [7]. The performance of the multi-agent systems is highly dependent on the ability of the agents to be able to localize themselves especially in areas where the absolute location of the agents is unavailable or inaccurate [6][8][9]. The knowledge of the position of the agents or nodes helps to improve the ability of the agents to effectively interact with the environment and the other nodes that are in the network thus enabling cooperative computing [10].

This aspect of localization is also extremely important when there is human-robot collaboration in undertaking the task, where the robot constantly required to know the position of the human target to effectively assist the human to undertake different operations [11].

The localization problem has been subject to extensive study, and different localization strategies have been developed. Some of these works are reviewed in the section.
1.1 Literature Review

It is desirable in a multi-agent system that each agent can be aware of its states and estimate the states of the other agents so that they can to successfully implement their tasks in a collaborative fashion [6]. Thus, different strategies have been put forward. This section will briefly elucidate the strategies and try to explain their strengths and weaknesses.

Most multi-agent systems rely on the use of Global Positioning System (GPS) which provides global absolute positioning information to the agents. Therefore, each agent is aware of its global position and can obtain the position of other agents through communication. The use of GPS requires that a direct line of sight (LOS) emanates from the satellites to the GPS receivers [12]. Since the technology has poor signal penetration capability, it hinders the consumer-grade GPS receivers from acquiring reliable readings in occluded areas like forests, indoors, and urban areas [13].

Moreover, the use of GPS systems in multi-agent systems require all-to-all communications between the agents which is very costly considering the constraints in the bandwidth of the communication channels thus rendering it impractical when implementing localization in a multi-agent system with a very large number of agents [14]. To eliminate the over-reliance on communication between the agents, it is desired that, the agents to use their on-board sensors to be able to localize other agents without using communication explicitly.

The introduction of high-performance sensors and processors has increased the interest in the field of simultaneous localization and mapping (SLAM) [15]. Since these sensors can be equipped in micro aerial vehicles, many works have been done where the Micro aerial vehicles perform SLAM. According to [16], and [17] the use of Visual Odometry (VO) to perform SLAM [18] was done by relying on pixel and features matching to obtain sparse maps. Other works [19] [20] propose methods that can produce real-time full dense maps. However, the visual SLAM algorithms
proposed in [19] and [20] impose a very high computational burden and are not suitable to implement in systems with limited computational capabilities, which is the case for most the companion computers used in mobile robots.

A Non-Iterative SLAM (NI-SLAM) algorithm introduced by [21] makes use of a single keyframe training to do visual data association and uses that data association to provide pose estimates. The implementation of the NI-SLAM solves the 6 Degree of Freedom pose estimation problem by decoupling the pose estimation problem into separate sub-spaces, separately solving the 3D motions and coupling back the solutions in the original systems [21][15]. This method was shown to reduce significantly the complexity of the pose estimation problem and thus making the NI-SLAM more suitable compared to the tradition visual SLAM in real-time implementation. Since NI-SLAM has a low computational requirement, it can be implemented in systems with limited computation capabilities [15].

Nevertheless, the NI-SLAM, though computationally light, is affected by visual drift over the long term. One of the algorithms that alleviate visual drift is the loop closure algorithm which uses visual techniques that require extra computational requirements or pre-trained data [22][23]. Thus, requiring high computational power.

Instead of relying on loop closure to eliminate the visual drift from NI-SLAM, [15] proposes an algorithm, that incorporates distance measurements done by Ultra-Wide Band (UWB) sensors and uses it to correct the visual drift. The fusion of the UWB sensors reading can be done using the Extended Kalman Filter (EKF), but [15] claims that the EKF suffers from measurement outliers for highly non-linear systems. To overcome this issue, a graph optimization approach was used to ensure good trajectory estimation. This method was reported to effectively eliminate visual drift and thus produce real-time dense drift-free maps [15].

The performance of the Global Positioning System (GPS) in multi-agent system can recuperated by the introduction of a cooperative localization approach which
makes use of the Ultra-Wide Band (UWB) \[6\][14][24]. In a cooperative localization approach, the agents use the relative measurements with respect to each other as a feedback signal to refine the joint team poses estimations [14][24]. The cooperative localization approach can be centralized or decentralized. In the centralized Cooperative localization, all the agents in the system communicate with the fusion center to exchange the distance measured between the agents. The fusion center accurately estimates the position of each agent and sends the estimates to each agent[14]. This centralized cooperative localization system is computationally very costly to the fusion center and also it requires a lot of communication between the fusion center and the agents which undesirable. Furthermore, the centralized system has a single point of failure and thus making it undesirable to be implemented in real system[14].

A more desirable system, the decentralized cooperative localization system is when there is no fusion center, each agent is localizing another agent by the use of its on-board sensors example UWB sensors and allow restricted communication between the agents so that they can exchange information about their distances between agents and cross-correlations to be used as feedback to obtain a refined joint team pose estimation [14][24]. The difficulty in implementing the decentralized cooperative localization system is how to design algorithms that maintain a reasonable communication and computational cost which will ensure consistency of the estimation process [14]. In [24], a partially decentralized system has been developed based on Extended Kalman Filter (EKF) which had a good performance even in the presence of unreliable communication links between the agents.

Due to the aforementioned reasons, the UWB sensor technology surpasses the GPS technology in proving reliable position measurements especially in occluded areas example indoors and forests [6]. This is because the UWB has high signal penetration power in presence of obstacles and accurate ranging due to fine delay resolution resulting from its large bandwidth [25][26][27]. UWB sensor technology
was used to develop accurate real-time indoor positioning systems which fuses the measurement from the nertial Measurement Unit (IMU) with the help of EKF to obtain a very accurate position [28][1].

Furthermore, [29] developed a robust robot peer-to-peer localization system which made measurements done by the use of three UWB anchors attached to a moving robot called an anchor to localize another moving robot called tag. The anchor made use of the EKF to obtain a good estimate of relative position which was used as a feedback in the formation control closed-loop system.

1.2 Objectives and Contributions

In the literature review, different localization strategies that use different sensor technologies have been discussed. The strengths and weaknesses of some of these strategies have been pointed out with possible solutions on how to overcome the weaknesses. But most of these strategies do not take advantage of the behavior of the agents. They simply rely on the sensor measurements and make use of filtering techniques like EKF to obtain the estimate of the states of the other agents [28][29].

In multi-agent systems, the agents in the teamwork in close vicinity to perform their tasks. And as they are working together they move together as a swarm [8]. It will be interesting if the localization algorithm tries to exploits this behavior to obtain more accurate state estimates.

This thesis tries to illustrate how to make use of the prior information on how the agents are behaving to our advantage to refine our estimates of the states of the agents. In particular, the thesis demonstrates how well will the estimation of the states of the agents be, if the localization scheme reflects the fact that the agents are in swarming behavior when the agents are exhibiting a swarming behavior.

This work mainly aspires to develop a novel localization strategy in multi-agent systems that exploits the swarming behavior of the agents to obtain a more accurate
estimate of the states of the agents.

This thesis resulted in contributions that are outlined in the following streams:

• To develop a formation control strategy for the multi-agents to demonstrate swarming behavior.

• To develop a real-time decentralized localization algorithm that exploits the swarming behavior of the agents.

• To compare the performance of the developed strategy with the real-time decentralized localization algorithm which does not exploit the swarming behavior of the agents.

1.3 Thesis Outline

To accomplish the goals set in this thesis, Chapter 2 will bring to light a review about the modeling of quadrotor which will be used in this work to represent the agents, Chapter 3 will demonstrate the development of position control and formation control algorithm, Chapter 4 will focus on developing the localization algorithms and analyze their performance in simulations, Chapter 5 discusses the implementation of control strategies and localization algorithms developed in chapter 3 and 4, respectively, on the experimental setup. Chapter 6 will discuss the obstacle avoidance algorithms used in the European Robotics League (ERL) that took place in Spain.
Chapter 2

Modeling of a Quadrotor

A quadcopter is an Unmanned Aerial vehicle that contains four propellers as the source of its thrust [30]. Recently, the use of quadcopters has become ubiquitous for different purposes. Mostly, quadcopters can be used for security and surveillance, for research purposes and other advanced purposes like the 3D map generation [30][31].

A quadcopter contains four propellers and therefore it has four inputs. And, the control of the quadcopter involves 12 state equations and therefore the quadcopter is highly underactuated [31]. State coupling and nonlinear state equations make the control of a quadcopter a very complex and interesting problem for the control engineers [32]. Currently, many researchers have dedicated their time to find different ways in which a quadcopter can be controlled [30][31].

The kinematic analysis of the quadcopter will be introduced, where different variables and representations of important parameters will be explained. Furthermore, the relationship that exists between different parameters will be demonstrated.

Finally, a kinetic analysis will be performed where we will derive the equations of linear motions by the use of Newtons second law of motion and the equations of the rotating body will also be derived by the use of Euler-Newton equations of a rotating body.

MATLAB and Simulink environment, initially developed by [33], was used to simulate kinematics and dynamic equation developed in this chapter and later on the simulation will be used to analyze the response of the position and formation control algorithms based on PI controller which will be explained in the next chapter.
2.1 Kinematics of a Quadcopter

Due to the complexity of controlling six degrees of freedom, certain notations are developed to facilitate the representation of certain critical variables. As an example

\[ \mathbf{v}_{b|i} \]  

(2.1)

Where, in the expression (2.1), \( \mathbf{v} \) is the base variable which can be linear acceleration, linear velocity or any other vector. Since most of the vectors like velocity are relative quantities, the lower subscript \( b|i \) shows that the variable is a relative velocity of the coordinate system \( b \) with respect to coordinate system \( i \). And the variable is represented as components of coordinate system \( b \) as indicated by the superscript \( b \).

The study of the kinematic model involves examining and analyzing the motion of the quadcopter without considering the forces and torques subjected to it [34].

![Figure 2.1: Kinematics of Quadcopter [1].](image)

To study the motion of the quadcopter robot, it is assumed that there is an inertial coordinate system which is denoted by \( i \) is assumed to be fixed on the surface of the Earth [31]. For practical purposes, we will assume that the Earths surface is fixed and thus we will ignore any acceleration caused by the rotation of the Earth since the
angular acceleration will be very small to be put into account [32].

The Inertial coordinate system is a right handed coordinate system i.e. it obeys the right hand rule of cross multiplication with coordinate axis $OX^i$, $OY^i$, and $OZ^b$ which are mutually perpendicular to each other. $i$ is a unit vector in $OX^i$ direction, $j$ is unit vector in $OY^i$ direction and $k$ is unit vector in the $OZ^b$ direction.

Another coordinate system which is attached at the center of mass of the quadcopter is assumed. This coordinate system will be moving with the quadcopter. It will have $OX^b$ along the arm of motor 1 that spins in an anticlockwise direction. The $OY^b$ axis will be along the arm of motor 2 which spin in clockwise direction and $OZ^b$ will point upwards [31][33]. The unit vectors along the $OX^b$, $OY^b$, and $OZ^b$ directions are $i^b$, $j^b$, and $k^b$, respectively. This configuration is called the plus configuration (+). There is another configuration called the cross configuration (x). In this work, we will only use the plus configuration due to its simplicity.

The position vector of the quadcopter’s center of mass relative to the inertial frame projected on the inertial frame is a vector which extends from the origin of the inertial coordinate system to the origin of the body coordinate system [32]. This vector shows the position of the quadcopter at any moment. The vector is denoted as $r_{bi}^i$. The position vector is expressed as follows.

$$r_{bi}^i = Xi + Yj + Zk$$

(2.2)

Where in equation (2.2), $X$ is the position vector component of the center of mass of the quadcopter in the X-axis of the inertial frame ($OX^i$), $Y$ is the position vector component of the center of mass of the quadcopter in the Y-axis of the inertial frame ($OY^i$) and $Z$ is the position vector component of the center of mass of the quadcopter in the Z-axis of the inertial frame ($OZ^i$) [31].

The quadcopter will have a linear velocity with respect to the inertial coordinate
The velocity of the quadcopter is expressed in terms of components of the body frame, $v^b_{bi}$. It is easy to confuse people that how comes the quadcopter will have velocity in the body frame since it is always fixed on a quadcopter and hence the relative velocity of a quadcopter with respect to its body coordinate system is zero. But $v^b_{bi}$ is the velocity of the quadcopter with respect to the inertial frame (not the body frame) expressed as components of the body frame [32]. The linear velocity is expressed as follows.

$$v^b_{bi} = U^b_i + V^b_j + W^b_k$$  \hspace{1cm} (2.3)

Where $U$, $V$, and $W$ are components of the total linear velocity along X axis ($OX^b$), Y axis ($OY^b$), and Z axis ($OZ^b$) of the body frame, respectively.

Apart from having the linear velocity, a quadcopter will have rotational motion. The angular velocity of the quadcopter with respect to inertial frame whose components are projected on the body frame is denoted as $\omega^b_{bi}$ is expressed as follows [32].

$$\omega^b_{bi} = P^b_i + Q^b_j + R^b_k$$  \hspace{1cm} (2.4)

Where $P$, $Q$, and $R$ are components of total angular velocity taken along X axis ($OX^b$), Y axis ($OY^b$) and Z axis ($OZ^b$) of the body frame, respectively [33].

2.1.1 Quadcopter Attitude with respect to the Inertial Frame

There are two coordinate systems which are the inertial coordinate system which is fixed to the Earth and the body coordinate system which is free to move and rotate with respect to the inertial coordinate system [32] [33]. Some angles are used to model the rotation of the two coordinate systems [31] [33]. These angles are called the Eulers
Angles. The rotation of the body frame to the inertial frame can be decomposed into three sequential simple rotations called the ZYX sequence of rotation \([32]\). In the ZYX sequence of rotation, the body is first rotated about the Z-axis, and then rotated about the Y-axis and finally, it is rotated about the X-axis \([31]\). These rotations will be explained briefly as follows.

**Rotation about the Z-axis**

This is the rotation of the quadcopter from the inertial frame to the intermediate frame 1 about the \(OZ^i = OZ^1\) with a heading or yaw angle \(\psi\) as the rotation angle \([32]\). In this transformation, points of on the quadcopter are rotated from the inertial frame to the intermediate frame 1 by the use of transformation matrix \(C^1_i\) \([32]\). This is shown mathematically as follows

\[
\begin{bmatrix}
  x^1 \\
  y^1 \\
  z^1
\end{bmatrix}
= C^1_i
\begin{bmatrix}
  x^i \\
  y^i \\
  z^i
\end{bmatrix}
\]

(2.5)

\[
\begin{bmatrix}
  x^1 \\
  y^1 \\
  z^1
\end{bmatrix}
= \begin{bmatrix}
  \cos \psi & \sin \psi & 0 \\
  -\sin \psi & \cos \psi & 0 \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x^i \\
  y^i \\
  z^i
\end{bmatrix}
\]

(2.6)

The angular velocity of the frame 1 relative to the inertial frame is

\[
\omega^1_i = \omega^i_i = \begin{bmatrix}
  0 \\
  0 \\
  \frac{d\psi}{dt}
\end{bmatrix}
\]

(2.7)
**Rotation about the Y axis**

This is the rotation of the quadcopter from the intermediate frame 1 to the intermediate frame 2 about the \( OY^1 = OY^2 \) with a pitch angle \( (\theta) \) as the rotation angle \(^{32}\). In this transformation, points on the quadcopter are rotated from the inertial frame to the intermediate frame 1 by applying the transformation matrix \( C^2_1 \) \(^{33}\). This is shown mathematically as follows

\[
\begin{bmatrix}
x^2 \\
y^2 \\
z^2
\end{bmatrix} = C^2_1
\begin{bmatrix}
x^1 \\
y^1 \\
z^1
\end{bmatrix}
\]

\[ (2.8) \]

\[
\begin{bmatrix}
x^2 \\
y^2 \\
z^2
\end{bmatrix} =
\begin{bmatrix}
\cos \theta & 0 & -\sin \theta \\
0 & 1 & 0 \\
\sin \theta & 0 & \cos \theta
\end{bmatrix}
\begin{bmatrix}
x^1 \\
y^1 \\
z^1
\end{bmatrix}
\]

\[ (2.9) \]

The angular velocity of the frame 2 relative to the frame 1 is

\[
\omega^1_{2|1} = \omega^2_{2|1} = \begin{bmatrix}
0 \\
\frac{d\theta}{dt} \\
0
\end{bmatrix}
\]

\[ (2.10) \]

**Rotation about the X axis**

This is the rotation of the quadcopter from the intermediate frame 1 to the intermediate frame 2 about the \( OY^1 = OY^2 \) with a rolling angle \( (\phi) \) as the rotation angle \(^{32}\). In this transformation points on the quadcopter are rotated from the inertial frame to the intermediate frame 1 by employing the transformation matrix \( C^b_2 \) \(^{33}\).
This is shown mathematically as follows

\[
\begin{bmatrix}
  x^b \\
y^b \\
z^b
\end{bmatrix}
\begin{bmatrix}
x^2 \\
y^2 \\
z^2
\end{bmatrix}
= C_b^2
\]

(2.11)

\[
\begin{bmatrix}
x^b \\
y^b \\
z^b
\end{bmatrix}
= \begin{bmatrix}
1 & 0 & 0 \\
0 & \cos \varphi & \sin \varphi \\
0 & -\sin \varphi & \cos \varphi
\end{bmatrix}
\begin{bmatrix}
x^2 \\
y^2 \\
z^2
\end{bmatrix}
\]

(2.12)

The angular velocity of the frame b relative to the frame 2 is

\[
\omega_{b/2}^b = \omega_{b/2}^2 = \begin{bmatrix}
\frac{d\varphi}{dt} \\
0 \\
0
\end{bmatrix}
\]

(2.13)

To rotate a vector from the inertial frame to the body frame, then one has to transform from inertial frame to the intermediate frame 1, from the intermediate frame 1 to the intermediate frame 2 and finally from the intermediate frame 2 to the body frame [32]. And therefore, a transformation from the inertial to the body frame is equivalent to the multiplication of the three sequential transformations as shown below.

\[
C_i^b = C_2^b(\varphi) \times C_1^2(\theta) \times C_1^i(\psi)
\]

(2.14)
\[
C^b_i = \begin{bmatrix}
1 & 0 & 0 \\
0 & \cos \varphi & \sin \varphi \\
0 & -\sin \varphi & \cos \varphi \\
\end{bmatrix} \begin{bmatrix}
\cos \theta & 0 & -\sin \theta \\
0 & 1 & 0 \\
\sin \theta & 0 & \cos \theta \\
\end{bmatrix} \begin{bmatrix}
\cos \psi & \sin \psi & 0 \\
-\sin \psi & \cos \psi & 0 \\
0 & 0 & 1 \\
\end{bmatrix}
\tag{2.15}
\]

After performing the matrix multiplication you will obtain the following transformation matrix \([32]\).

\[
C^b_i = \begin{bmatrix}
\cos \theta \cos \psi & \cos \theta \sin \psi & -\sin \theta \\
\sin \varphi \sin \theta \cos \psi - \cos \varphi \sin \psi & \sin \varphi \sin \theta \sin \psi + \cos \varphi \cos \psi & \sin \varphi \cos \theta \\
\cos \varphi \sin \theta + \sin \varphi \sin \psi & \cos \varphi \sin \theta \sin \psi - \sin \varphi \cos \psi & \cos \varphi \cos \theta \\
\end{bmatrix}
\tag{2.16}
\]

The above transformation matrix, \(C^b_i\), in \((2.16)\) can be used to transform a vector which is in the inertial frame to equivalent vector in the body frame \([35]\). This can be shown by the following equation.

\[
\begin{bmatrix}
x^b \\
y^b \\
z^b \\
\end{bmatrix} = C^b_i \begin{bmatrix}
x^i \\
y^i \\
z^i \\
\end{bmatrix}
\tag{2.17}
\]

It is also possible to rotate a vector that is in the body frame back to the inertial frame by using the transformation matrix \(C^b_i\) as shown in the following equation \([35][32]\).

\[
\begin{bmatrix}
x^i \\
y^i \\
z^i \\
\end{bmatrix} = C^i_b \begin{bmatrix}
x^b \\
y^b \\
z^b \\
\end{bmatrix}
\tag{2.18}
From the two equations it can be shown that $C_i^b$ and $C_b^i$ are inverse of each other. It follows from the properties of transformations that the inverse of a transformation matrix is the same as the transpose of the transformation matrix $C_i^b$.

$$C_i^b = (C_i^b)^{-1} = (C_i^b)^T \quad (2.19)$$

$$C_b^i = \begin{bmatrix}
cos \theta \cos \psi & \sin \varphi \sin \theta \cos \psi - \cos \varphi \sin \psi & \cos \varphi \sin \theta + \sin \varphi \sin \psi \\
cos \theta \sin \psi & \sin \varphi \sin \theta \sin \psi + \cos \varphi \cos \psi & \cos \varphi \sin \theta \sin \psi - \sin \varphi \cos \psi \\
-\sin \theta & \sin \varphi \cos \theta & \cos \varphi \cos \theta
\end{bmatrix} \quad (2.20)$$

Now, the angular velocity is expressed as components of the body frame so that, the relationship between the rate of change of the Eulers angles and components of total angular velocity in the body frame can be found.

$$\omega_b^{i} = \omega_1^{i} + \omega_2^{b} + \omega_3^{b} \quad (2.21)$$

But it is known that

$$\omega_1^{i} = C_2^{b} \times C_1^{2} \times \omega_1^{b} \quad (2.22)$$

$$\omega_1^{i} = C_2^{b} \times \omega_2^{b} \quad (2.23)$$
Substituting (2.22) and (2.23) in (2.21)

\[ \omega_{b|i}^b = C_{2}^b \times C_{2}^1 \times \omega_{1|i}^1 + C_{2}^b \times \omega_{2|i}^2 + \omega_{b|i}^b \] (2.24)

Written in matrix notations as

\[
\begin{bmatrix}
1 & 0 & -\sin \theta \\
0 & \cos \varphi & \sin \varphi \cos \theta \\
0 & -\sin \varphi & \cos \varphi \cos \theta
\end{bmatrix}
\begin{bmatrix}
\dot{\phi} \\
\dot{\theta} \\
\dot{\psi}
\end{bmatrix}
\] (2.25)

By comparing the (2.13) and (2.25), we obtain the following equation

\[
\begin{bmatrix}
P \\
Q \\
R
\end{bmatrix} =
\begin{bmatrix}
1 & 0 & -\sin \theta \\
0 & \cos \varphi & \sin \varphi \cos \theta \\
0 & -\sin \varphi & \cos \varphi \cos \theta
\end{bmatrix}
\begin{bmatrix}
\dot{\phi} \\
\dot{\theta} \\
\dot{\psi}
\end{bmatrix}
\] (2.26)

2.2 Dynamics of a Quadcopter

Having seen the kinematic analysis of a quadcopter now we are in the right position to discuss the kinetics or dynamics of a quadcopter. Kinetics is the analysis of quadcopter by considering the forces and torques and their relationship with the motion by applying Newtons laws of motion [30][31][32]. To apply Newtons law of motion we have to consider all the external forces and torques acting on a body of the quadcopter [34]. And later we will add the forces and moments to obtain the resultant linear and angular accelerations.

The following is the brief discussion of the forces and torques acting on the body of the quadcopter.
2.2.1 Forces and Torques acting on the body of a Quadcopter

Thrust of Propellers

The thrust force acting on the propellers of a quadcopter is the forces that provide the lift force of a quadcopter \[30\]. The electric energy stored in the battery is converted into rotational mechanical energy by the motors which are connected to the propellers. While the propellers are rotating in the air, there occur aerodynamic forces which have two components, the lift and the drag \[32\] \[33\]. The lift is the component of the aerodynamic force that acts perpendicular to the motion of the propellers and it is the one responsible for the upward motion of the quadcopter \[36\]. According to \[31\] and \[32\] the lift force acting on the propellers is given by the following formulation

\[ T_n = C_T \rho A_r r^2 \omega_n^2 \]  

(2.27)

Where, \( C_T \) is the thrust coefficient for the specific propeller \[32\], \( \rho \) is the density of air \[31\], \( A_r \) the area swept of the propellers rotation, \( r \) is the distance from the center of the propeller to the tip of the propeller and \( \omega_n \) is the angular velocity of the nth motor in rpm and the case of quadcopter n can be 1,2,3 or 4.

But for simplicity, we lump up all the constant parameters in one constant variable as shown in the equation below. Assuming that the density of air does not change significantly. We can take it as constant

\[ K_f = C_T \rho A_r r^2 \]  

(2.28)

Substituting (2.28) in (2.27), we will obtain

\[ T_n = K_f \omega_n^2 \]  

(2.29)
The value of $K_f$ can be obtained by doing calculations from above equation (2.28) or it can be obtained experimentally [33]. The total Thrust exerted on the body of the quadrotor is calculated as follows [32] [33].

$$T_{Bz} = T_1 + T_2 + T_3 + T_4$$

(2.30)

Substituting the equation (2.29) in (2.30)

$$T_{Bz} = K_f\bar{\omega}_1^2 + K_f\bar{\omega}_2^2 + K_f\bar{\omega}_3^2 + K_f\bar{\omega}_4^2$$

(2.31)

Writing it in matrix format

$$T_B = \begin{bmatrix} 0 \\ 0 \\ K_f\bar{\omega}_1^2 + K_f\bar{\omega}_2^2 + K_f\bar{\omega}_3^2 + K_f\bar{\omega}_4^2 \end{bmatrix}$$

(2.32)

Torque of Propellers

The rotation of the propellers not only causes the lift force, but also they cause moments or torques in the z-direction of the body frame, $OZ^b$. It was discussed in [31] and [32] that the rotation of the propellers results in the occurrence of the aerodynamic force which has two components, the lift component, and the drag. The drag component acts in the parallel and opposite direction to the motion of the propeller [32]. The drag component of the aerodynamic force is responsible for the formation of torque due to the propellers on the body of the quadcopter robot [31] [30]. According to [31] and [32] the moment caused by the propellers due to drag force is given by the following formulation.

$$M_n = (-1)^n \frac{1}{2} \rho AC_D r^3 \bar{\omega}_n^2$$

(2.33)
Where, $M_n$ is the moment caused by nth motor, $\rho$ is density of the air, $A$ is the frontal area of the propeller given by the multiplication of chord length and maximum thickness of the propeller, $C_D$ is a drag coefficient of the propeller [32], $r$ is the distance from the center of the propeller to the tip of the propeller [33] and $\omega_n$ is the angular velocity of the nth motor in rpm and in the case of quadcopter n can be 1, 2, 3 or 4.

It can also be observed that there is a factor of $(-1)^n$ in the equation (2.33). This factor is added due to Newton’s third law of motion. Since when the propeller is rotating in an anticlockwise direction, it will result in a clockwise moment on the propeller and vice versa. Note that motor 1 is rotating in the anticlockwise direction so it will result in a clockwise moment on the body. Assuming that the density of air remains constant. We can lump up all the constants together to obtain one constant variable as shown below

$$K_m = \frac{1}{2} \rho A C_D r^3$$  \hspace{1cm} (2.34)

After substitution of equation (2.34) in (2.33)

$$M_n = (-1)^n K_m \omega_n^2$$  \hspace{1cm} (2.35)

The value of $K_m$ can be obtained from constants in the equation (2.33) or it can be obtained experimentally [33].

2.2.2 Gyroscopic Moments

Gyroscopic moments of the motors is a physical effect in which the gyroscopic torques tends to align the spin axis of the motor along the inertial z-axis [32] [33]. The gyroscopic moment is obtained from the following formulation
\[ M_g = \omega_{bi}^b \times \begin{bmatrix} 0 \\ 0 \\ J_r \Omega_r \end{bmatrix} \]  \hspace{1cm} (2.36)

Where, \( M_g \) is the gyroscopic moment, \( J_r \) is a moment of the motor, \( \Omega_r \) is the summation of all the motor rotational speed in rpm.

According to \cite{32} \cite{33}, the value of \( \Omega_r \) is obtained by the following formula

\[ \Omega_r = \bar{\omega}_1 - \bar{\omega}_2 + \bar{\omega}_3 - \bar{\omega}_4 \]  \hspace{1cm} (2.37)

The final expression of the gyroscopic moment after computing the cross product is given by

\[
M_g = \begin{bmatrix}
Q J_r \Omega_r \\
-P J_r \Omega_r \\
0
\end{bmatrix}
\]  \hspace{1cm} (2.38)

**Moments Acting on a Quadcopter Due To Thrust And Torque of Propellers**

The total moment acting in the X-direction of the body frame (\( OX^b \)) is given by the following relation \cite{31} \cite{32}.

\[ M_x = T_2d - T_4d \]  \hspace{1cm} (2.39)
Substituting equation (2.29) in equation (2.39), we will have

\[ M_x = dK_f \omega_2^2 - dK_f \omega_4^2 \]  \hspace{1cm} (2.40)

Where \( d \) is defined as the distance from the center of the hub to the motor [33].

The total moment acting in the \( Y \) direction of the body frame (\( OY^b \)) is given by the following relation [31] [32].

\[ M_y = -T_1 d + T_3 d \]  \hspace{1cm} (2.41)

Substituting equation (2.29) in equation (2.41), we will have

\[ M_y = -dK_f \omega_1^2 + dK_f \omega_3^2 \]  \hspace{1cm} (2.42)

Where \( d \) is defined as the distance from the center of the hub to the motor [33].

The total moment acting in the \( Z \) direction of the body frame (\( OZ^b \)) is given by the following relation [31] [32].

\[ M_z = -M_1 + M_2 - M_3 + M_4 \]  \hspace{1cm} (2.43)

Substituting equation (2.35) in (2.43), we will have

\[ M_z = -K_m \omega_1^2 + K_m \omega_2^2 - K_m \omega_3^2 + K_m \omega_4^2 \]  \hspace{1cm} (2.44)
2.3 Translation Equations of Motion

Applying Newtons equation of motion on the body frame as shown below. According to [32] we have

\[ m \ddot{\mathbf{v}}_b = T_B + C_i \]

\[
\begin{bmatrix}
0 \\
0 \\
mg
\end{bmatrix}
\]  \hspace{1cm} (2.45)

But the change in velocity is a complex problem it involves a change in the magnitude of the velocity and the change in the direction of the velocity [31]. According to Coriolis law [31],

\[ \dot{\mathbf{v}}_b = \frac{d}{dt} (\mathbf{v}_b) + \omega_b \times \mathbf{v}_b \]  \hspace{1cm} (2.46)

After manipulating the equation (2.46) the resultant equation is

\[
\begin{bmatrix}
\dot{U} \\
\dot{V} \\
\dot{W}
\end{bmatrix} = \begin{bmatrix}
WQ - VR \\
UR - WP \\
VP - UQ
\end{bmatrix}
\]  \hspace{1cm} (2.47)

Substituting equations (2.32) and (2.47) in (2.45)

\[ m \begin{bmatrix}
\dot{U} \\
\dot{V} \\
\dot{W}
\end{bmatrix} + \begin{bmatrix}
WQ - VR \\
UR - WP \\
VP - UQ
\end{bmatrix} = \begin{bmatrix}
0 \\
0 \\
K_f \omega_1^2 + K_f \omega_2^2 + K_f \omega_3^2 + K_f \omega_4^2
\end{bmatrix} + \begin{bmatrix}
mg \sin \theta \\
-mg \cos \theta \sin \varphi \\
-mg \cos \theta \cos \varphi
\end{bmatrix} \]

\hspace{1cm} (2.48)
After some simple algebraic manipulations, the resultant equation will be

\[
\begin{bmatrix}
\dot{U} \\
\dot{V} \\
\dot{W}
\end{bmatrix} = \frac{1}{m} \begin{bmatrix}
0 & 0 & K_f \omega_1^2 + K_f \omega_2^2 + K_f \omega_3^2 + K_f \omega_4^2 \\
0 & g \sin \theta & -g \cos \theta \sin \varphi \\
K_f \omega_1^2 + K_f \omega_2^2 + K_f \omega_3^2 + K_f \omega_4^2 & -g \cos \theta \sin \varphi & -g \cos \theta \cos \varphi
\end{bmatrix} = \begin{bmatrix}
WQ - VR \\
UR - WP \\
VP - UQ
\end{bmatrix}
\]

(2.49)

From the above equation, we obtain the acceleration of the quadcopter written as the components of the body frame. The velocity of the quadcopter projected in the inertial frame is obtained by rotating the velocity vector \( \mathbf{v}_{b|i} \) from the body frame to the inertial frame by the use of the transformation matrix \( C_{b|i} \)

\[
\mathbf{v}_{b|i} = C_{b|i} \mathbf{v}_{b|i}
\]

(2.50)

Substituting the rotational matrix from equation (2.20) in (2.50)

\[
\begin{bmatrix}
\dot{X} \\
\dot{Y} \\
\dot{Z}
\end{bmatrix} = \begin{bmatrix}
\cos \theta \cos \psi & \sin \varphi \sin \theta \cos \psi - \cos \varphi \sin \psi & \cos \varphi \sin \theta + \sin \varphi \sin \psi \\
\cos \theta \sin \psi & \sin \varphi \sin \theta \sin \psi + \cos \varphi \cos \psi & \cos \varphi \sin \theta \sin \psi - \sin \varphi \cos \psi \\
-\sin \theta & \sin \varphi \cos \theta & \cos \varphi \cos \theta
\end{bmatrix} \begin{bmatrix}
U \\
V \\
W
\end{bmatrix}
\]

(2.51)
2.4 Rotational Equations of Motion

According to [34], the momentum $H$ of a quadcopter is obtained from the following equation

$$
H = \int_{i}^{b} \left[ (x^2 + y^2) P - xyQ - xzR \right] dm + \int_{j}^{b} \left[ (z^2 + x^2) Q - yzR - xyP \right] dm \\
+ \int_{k}^{b} \left[ (x^2 + y^2) R - xzP - yzQ \right] dm
$$

Moment of inertia in x direction is given by [34]

$$
I_{xx} = \int (y^2 + z^2) \, dm
$$

Also moment of inertia in y direction [34]

$$
I_{yy} = \int (x^2 + z^2) \, dm
$$

And moment of inertia in z direction [34]

$$
I_{zz} = \int (x^2 + y^2) \, dm
$$

Products of moments in x and y direction [34]

$$
J_{xy} = \int (x y) \, dm
$$

Products of moments in x and z direction [34]

$$
J_{xz} = \int (x z) \, dm
$$
Products of moments in y and z direction

\[ J_{yz} = \int (yz) \, dm \]  \hspace{1cm} (2.58)

Due to the symmetry of the quadcopter, the products of moments are zero

\[ J_{xy} = J_{xz} = J_{yz} = 0 \]  \hspace{1cm} (2.59)

Therefore the equation to calculate the momentum (2.52) of the quadcopter simplifies to become after substituting (2.59):

\[ \mathbf{H} = P I_{xx} \mathbf{i}^b + Q I_{yy} \mathbf{j}^b + R I_{zz} \mathbf{k}^b \]  \hspace{1cm} (2.60)

It follows from Newton's law of motion, that the rate of change of angular momentum is equal to the summation of all external moments acting on the body [36][31][32].

\[ \dot{\mathbf{H}} = \begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} + \mathbf{M}_g \]  \hspace{1cm} (2.61)

But the quadcopter undergoes complex motion, the momentum undergoes both changes in magnitude and direction. Therefore, the rate of change of momentum is equal to the rate of change of magnitude and direction of momentum [32]

\[ \dot{\mathbf{H}} = \frac{d\mathbf{H}}{dt} + \omega_{bi}^b \times \mathbf{H} \]  \hspace{1cm} (2.62)
Where $\frac{dH}{dt}$ is the rate of change of magnitude of momentum and $\mathbf{\omega}_{bi} \times \mathbf{H}$ is a cross product in which the rate of change direction of momentum is obtained.

But

$$\frac{dH}{dt} = \dot{P}I_{xx} \mathbf{b} + \dot{Q}I_{yy} \mathbf{b} + \dot{R}I_{zz} \mathbf{b}$$  \hspace{1cm} (2.63)$$

And

$$\mathbf{\omega}_{bi} \times \mathbf{H} = QR(I_{zz} - I_{yy}) \mathbf{b} + PR(I_{xx} - I_{zz}) \mathbf{b} + PQ(I_{zz} - I_{xx}) \mathbf{b}$$  \hspace{1cm} (2.64)$$

Therefore equations (2.61) and (2.62) become

$$\frac{dH}{dt} + \mathbf{\omega}_{bi} \times \mathbf{H} = \begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} + Mg$$  \hspace{1cm} (2.65)$$

Which after substituting equations (2.63) and (2.64) and doing simple algebraic manipulations, the resultant expression is

$$\begin{bmatrix} \dot{P} \\ \dot{Q} \\ \dot{R} \end{bmatrix} = \begin{bmatrix} \frac{(dK_f \omega_z^2 - dK_f \omega_y^2 + Q \Omega_y - QR(I_{zz} - I_{yy}))}{I_{xx}} \\ \frac{(-dK_f \omega_y^2 + dK_f \omega_z^2 - P \Omega_x - PR(I_{xx} - I_{zz}))}{I_{yy}} \\ \frac{(-K_m \omega_z^2 + K_m \omega_x^2 - K_m \omega_y^2 + K_m \omega_z^2 - PQ(I_{zz} - I_{xx}))}{I_{zz}} \end{bmatrix}.$$  \hspace{1cm} (2.66)$$

All the above equations that were derived constitute the dynamic equations of the quadcopter. These equations were implemented using MATLAB and Simulink program to simulate the model of a quadcopter in which, the position control and formation control algorithms are going to be designed using the simulation before being implemented in the real system. Refer to Appendix A to see the model of the quadcopter which was initially developed by [33] and this thesis has built on top of
it in the design of the position and formation control algorithms.
Chapter 3

Position and Formation Control of Quadcopter

Our objective is to develop a localization algorithm that accounts for swarming behavior of the agents found in a multi-agent system (in our case a team of drones). So, therefore, swarming is an important assumption and prerequisite for our localization algorithm to have an accurate performance.

This chapter is going to focus on designing a formation control algorithm which will make the drones to exhibit swarming characteristics. Yet, before the formation control algorithm design is successful, it is necessary to ensure the drone has a good position control algorithm.

The design of these algorithms is going to be done in MATLAB and Simulink model [33]. Having the Robot Operating System (ROS) framework at our disposal, the controllers designed in MATLAB are implemented using Python programming language and launched on a more realistic simulation called GAZEBO to double-check the performance of the control strategy before being uploaded on the real quadcopters [37].

3.1 Position Control of a Quadcopter

The position is one of the most important variables to be controlled. The success of any autonomous mission involves the control of position [30][31][32]. The control strategy involves finding the error in X and Y direction which is the difference between the desired position and the current position of the drone [33].
\[ e_x = X_{com} - X \] (3.1)

and

\[ e_y = Y_{com} - Y \] (3.2)

Where \( X_{com} \) and \( Y_{com} \) are the desired positions in the X and Y directions, respectively. \( X \) and \( Y \) are the current positions of a drone.

The PI control law is as follows;

\[ v_{i,x,c}^i = K_p e_x + K_i \int_{0}^{t} e_x dt \] (3.3)

and

\[ v_{i,y,c}^i = K_p e_y + K_i \int_{0}^{t} e_y dt \] (3.4)

Where the \( K_p \) and \( K_i \) are the proportional and integral gains of PI controller, \( v_{x,c}^i \) and \( v_{y,c}^i \) the velocity command of the drone in the X and Y direction, respectively, in the inertial coordinate system.

Since the velocity commands are computed in the inertial coordinate system, they should be transformed to the body coordinate system by multiplying them with the transformation matrix. Also since the pitch angle and the roll angle are approximately zero during the normal flight operations, the transformation matrix will be highly dominated by the yaw angle [31][33].
\[ v_{x,c}^b = v_{x,c}^i \cos \psi + v_{y,c}^i \sin \psi \]  \hspace{1cm} (3.5)

and

\[ v_{y,c}^b = v_{x,c}^i \cos \psi - v_{y,c}^i \sin \psi \]  \hspace{1cm} (3.6)

Where \( v_{x,c}^b \) and \( v_{y,c}^b \) are the velocity commands transformed in the body coordinate system of a drone, and \( \psi \) is the yaw angle of a drone.

The velocity commands obtained are the input to the inner control loops which in this work we will assume to be taken care of by the flight controller, which is the case on the real implementation of the system.

The control law which was discussed was implemented in MATLAB and Simulink models and the Figures 3.1 and 3.2 below illustrate the performance of the system when the values of \( K_p \) was 0.6 and \( K_i \) was 0.08
The control strategy was implemented in a python program and using the ROS framework it was launched into a quadcopter simulation in Gazebo. The performance of the position control using PI controller can be seen the Figures 3.3 and 3.4 below.
It should be noted that these graphs do not represent the best performance achievable using the PI controller but rather the PI controller gain were obtained by trial and error. But even with just using the trial and error to tune the PI values of the position controller we were able to obtain acceptable performance. But Optimization techniques can always be used to obtain the best PI values which minimize or maximizes a certain objective function.

3.2 Formation Control of a Quadcopter

The objective of the formation control is to make a leader-follower configuration. The leader drone would be controlled by a joystick and the follower drone would follow the leader drone wherever it is going. The control strategy will try to maintain a constant relative distance between the drones in x and y-direction. Thus once the follower is executing this algorithm, it will be exhibiting the swarming behavior. And thus the leader drone can take that into account in its localization strategy.

In this control strategy, we will try to control the distance between the drones using the PI controller so that the drones maintain a constantly desired distance.
The error is the distance between the drones is defined as the difference between the desired distance between the drones and the actual distance between the drones. Mathematically, this is expressed as follows:

\[ e_{dx} = d_x - \| X - \hat{x}_{t,k} \| \]  

(3.7)

And

\[ e_{dy} = d_y - \| Y - \hat{y}_{t,k} \| \]  

(3.8)

Where, \( e_{dx} \) and \( e_{dy} \) are the desired distances between the drones in X and Y directions, respectively, \( e_{dx} \) and \( e_{dy} \) are errors in the distances in X and Y directions, respectively, \( \hat{x}_{t,k} \) and \( \hat{x}_{t,k} \) are estimates of X and Y position of the tag obtained either from the localization system (example Mocap system) or from the localization algorithm implemented on the anchor drone (the leader drone with the on-board sensors).

The control law is as follows.

\[
    v^i_{x,c} = - \text{sgn} (X - \hat{x}_{t,k}) \times \left( K_p e_{dx} + K_i \int_0^t e_{dx} dt \right)
\]

(3.9)

And

\[
    v^i_{y,c} = - \text{sgn} (Y - \hat{y}_{t,k}) \times \left( K_p e_{dy} + K_i \int_0^t e_{dy} dt \right)
\]

(3.10)

Where \( v^i_{x,c} \) and \( v^i_{y,c} \) are velocity command of the drone in X and Y direction respectively in the inertial coordinate system and \( K_p \) and \( K_i \) are the proportional
and integral gains of PI controller.

Velocity commands obtained are transformed in a body coordinate system using the equations (3.5) and (3.6) are fed into the on-board flight controller which takes care of the inner loops that control the attitude angles and velocities.

The control strategy is tested in the model developed in MATLAB and Simulink [33]. Where the leader was controlled to follow a diamond path and the follower drone (tag) while maintaining a constant distance of 2 meters from the leader. The following Figures 3.5 and 3.6 shows the performance of the drones in performing the aforementioned task.

![Figure 3.5: Formation control in X using PI controller in MATLAB/Simulink](image-url)
The control law was also implemented in Python programs. Using the ROS framework the programs were uploaded to two simulation drones (one leader and the other one follower) in Gazebo and the following Figures 3.7 and 3.8 show the performance of the controller in maintaining the desired distance between the drones which was set to be 2 meters.

Figure 3.6: Formation control in Y using PI controller in MATLAB/Simulink

Figure 3.7: Formation control in X using PI controller in ROS and Gazebo
Figure 3.8: Formation control in Y using PI controller in ROS and Gazebo

It can be noted that despite the constant disturbances the control strategy has been able to maintain a distance between the two drones around 2 meters. This controller can be further be improved to ensure that it reduces the error from the desired distance between the drones by the use of optimization techniques. But for this thesis, this performance is just enough.
Chapter 4

Swarm Localization

In the Literature Review 1.1, different localization strategies have been discussed and with their strengths and weaknesses. It was mentioned that the UWB technology has offered a promising future to be used as a perfect substitute for the GPS technology, especially in the indoors scenarios where the GPS technology, be an unreliable source of localization [24][28][29].

This chapter will superficially introduce the UWB technology and point out the features that make it attractive for the localization application. Most importantly different ranging methods will be discussed used for the general wireless signal localization methods.

Different position estimation algorithms will be introduced which are already existing in the literature.

Finally, a localization algorithm that capitalizes on the swarming behavior of the agents (drones) is proposed whose performance is juxtaposed with localization algorithm which does not take into consideration the fact that the drones are swarming. The comparison will be done while the two agents are swarming.

4.1 Introduction to Ultra-Wide Band (UWB) Technology

As the name suggests, UWB radio technology transmits signals with very high bandwidth [4]. UWB is radio technologies with a center frequency higher than 2.5GHz and has bandwidth larger than 500 MHz. Alternatively, another way to define UWB
radio technology is radio signals having fractional bandwidth which is higher than 0.2
with a center frequency lower than 2.5 GHz [28]. The bandwidth of a wireless sig-
nal is difference between the lower cut-off frequency and the higher cut-off frequency
[28][38][39] expressed mathematically as follows

\[ B = f_H - f_L \]  \hspace{1cm} (4.1)

Where \( B \) is the bandwidth of a signal, \( f_H \) is the high cut-off frequency, and \( f_L \) is the
lower cutoff frequency.

The cut-off frequency is the frequency limit beyond which the energy passing
through the system is attenuated instead of passing through the system without
attenuation [38][40]. Practically, the Cutoff frequency can be determined as the fre-
quency in which the power density of a signal is less than the half the Maximum
power density [4][41]. Center frequency is the frequency where the power density
is maximum [4][28][39]. Practically, this corresponds to the midpoint between the
higher cutoff frequency and lower cutoff frequency [39][41] expressed mathematically
as follows

\[ f_c = \frac{f_L + f_H}{2} \]  \hspace{1cm} (4.2)

Whereby \( f_c \) represents the center frequency, \( f_H \) and \( f_L \) is the high cutoff frequency and
low cutoff frequency respectively. Fractional bandwidth is the bandwidth of a device
divided by its center frequency [28][4][38]. Mathematically, fractional bandwidth is
calculated using the following equation

\[ B_{frac} = \frac{B}{f_c} = \frac{2(f_H - f_L)}{f_L + f_H} \]  \hspace{1cm} (4.3)

From the Shannon-Hartley theorem, the relationship between the channel capac-
ity, a bandwidth of a signal, and signal to noise ratio are obtained \[38\][40]. This relationship is stated using the following equation

\[
C = B \log_2(1 + SNR)
\]  

(4.4)

Where \(C\) is the channel capacity expressed in bits per second, \(B\) is the bandwidth of the channel and \(SNR\) is the signal to noise ratio.

As a result of this equation (4.4) above, it can be deduced that the Channel capacity is directly proportional to the bandwidth. And since the \(SNR\) is inside a logarithm function, it follows that channel capacity, \(C\) can be easily increased by increasing the bandwidth instead of increasing the \(SNR\) \[38\][40]. This allows the UWB sensor technology to have high channel capacity despite using less power \[25\][28][4].

Despite having a good characteristic of having a high Bandwidth which allows UWB to have high channel capacity. UWB technology can be a great source of interference for some of the systems if it is used carelessly leading to failures in systems using other types of signals with narrow bands \[28\][1]. To avoid such scenarios, some regulatory bodies have enacted some regulations which limit the power of using the UWB technology to avoid interference of UWB signals with other signals. For instance, in the USA, they have the Federal Communication Commission (FCC) which limits the indoor usage of UWB under part 15 limits which sets the limit for the Equivalent Isotropically Radiate Power (EIRP) \[28\].

The following Figure 4.1 shows the comparison of the power density of UWB technology with other existing technologies \[3][2].
4.1.1 Characteristics of the UWB technology

According to [28][4], these are the important features of the UWB technology which makes it ideal for localization.

The most obvious feature is the high bandwidth compared to any other technologies[4]. As a consequence of having high bandwidth, UWB technology is ideal for usage in high-speed communication [28].

High-time resolution making it ideal choice for short distance positioning systems [6][28]. This characteristic also comes as a direct consequence of UWB technology having high bandwidth causing the short-life time in the UWB signals. Due to the high time resolution and short wavelength of the UWB technology makes it’s signal immune to multi-path effect and fading [28][4].

It is also possible to transmit UWB signals using lower carrier frequencies allowing the UWB signals to a path through obstacles thus making the localization system robust even when there is no direct line of sight between the anchors and the tag [6][28].

The UWB signal transmission is more secure since UWB signals have the structure as the noise signals making it difficult to do eavesdropping [28].
4.1.2 Comparison with Other Widely Used Ranging Technologies

The UWB signal technology has been compared to other localization technologies in terms of accuracy, precision, and costs in [4]. In their survey, [3] compared the resolution of different localization technologies when used in different scenarios which are laid down in a scale which includes the indoors, urban outdoors, and in rural and remote areas. The following Figure 4.2 summaries their findings [3].

![Resolution Comparison Between Positioning Technologies](image)

Figure 4.2: Resolution Comparison Between Positioning Technologies (From [3])

The above Figure 4.2 points out that the UWB technology operating in Angle of Arrival, Time difference of Arrival or round-trip Time of Flight surpass any other technology when it comes to producing high-resolution distance measurements, making it ideal for localization in the indoors and even outdoors scenarios over a short distance, since it is limited in power [4].

The Table 4.1 below which shows the comparison of different localization technologies further illustrates that the UWB sensor is superior to other popular localization technologies used in short distances since UWB has very high accuracy and precision [4][3][12].
Table 4.1: Comparison of short range Positioning Technology (From [4])

<table>
<thead>
<tr>
<th>Technology</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Complexity</th>
<th>Power</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>UWB</td>
<td>&lt; 30cm</td>
<td>99%</td>
<td>Application Based</td>
<td>30mW</td>
<td>&lt; 30cm</td>
</tr>
<tr>
<td>2.4 GHz Zigbee</td>
<td>&gt; 2m</td>
<td>Upto 99%</td>
<td>Low</td>
<td>20mW to 40mW</td>
<td>&lt; 30m</td>
</tr>
<tr>
<td>2.4 GHz WiFi</td>
<td>&gt; 2m</td>
<td>Depends on standards</td>
<td>High</td>
<td>500mW to 1W</td>
<td>&lt; 100m</td>
</tr>
<tr>
<td>IR</td>
<td>&lt; 1m</td>
<td>50% Within 1m</td>
<td>Medium to High</td>
<td>High Variation</td>
<td>20m to 30m</td>
</tr>
<tr>
<td>Bluetooth</td>
<td>2m</td>
<td>95% within 2m</td>
<td>Medium</td>
<td>60mW</td>
<td>30m to 50m</td>
</tr>
</tbody>
</table>

As indicated in Table 4.1 the range where the UWB can be used is low compared to other methods due to the limit of power imposed on the UWB technology by the regulatory bodies to avoid interference with other system operating at the same time [28][4]. Also, the complexity and cost of the devices using UWB technology are dependent on the type of application for the devices. For instance, a device that is intended to support multi-user communication using UWB sensors will have to make use of the state of the art methods to resolve the multipath problem energy [43].

Thus due to the aforementioned reasons the UWB sensor technology is more favorable compared to other ranging 103 technologies and UWB sensor technology was used in this work to provide accurate ranging measurements for the localization algorithms which will be discussed later on the chapter. Now that the ranging technology is selected it is appropriate to discuss the different methods used in the literature to determine the distance estimate between two nodes.

4.2 Radio Frequency Ranging Methodologies

This section will briefly discuss different methods used in measuring distance with the use of Radiofrequency wireless signals, in particular, the UWB. There are four main methods used to measure distance namely Signal Strength, Angle of Arrival,
Time of arrival and Time difference of arrival. Each of these methods will be briefly explained.

4.2.1 Signal Strength

In this method, the distance between the source node and the destination node is measured by analyzing the power of the received signal at the destination node[4][28]. This method assumes that the strength of the transmitted signal is known by the destination node at the time it is initially being transmitted. Also, the method assumes that there is a relationship between the distance and the strength of the signal [4].

However, this method of calculating distance results to high inaccuracy because the signal can be attenuated along the path due to other phenomena like fading, reflections and others [44]. Sophisticated strategies should be used to mitigate the losses caused by the path-losses before measuring the received signal power strength.

4.2.2 Angle of arrival

This method is used to measure the angle between the anchor and the tag node[28]. This angle is measured in the anchor node by the use of two antennas with a known distance apart to measure the time of flight of the signal to the tag node. The two antennas will measure two-time measurements and use that to determine the angle of the tag node which proportional to the time differences between the time of flights measured by the two antennas [4]. This angle measurement can be used with triangulation methods to obtain the position of the tag node.

But since this method uses the measurements of the time of flight, the performance of this method depends on how accurate the time of flights can be measured by the two antennas. And another factor which limits the performance of this method is the proximity of the antennas which results in the extremely small time difference (in
order of nanoseconds) which are very difficult to detect and thus requiring complex circuitry [4].

4.2.3 Time Difference of Flight

In this method, the source node transmits the signal to more than one destination node [4][28]. Each destination node measures the time of flight by subtracting the time in which the signal leaves the source node from the time in which the signal arrives at that particular node. Each destination node acts as an imaginary center of a circle. And the position of the source node is obtained by the intersection of all the circles [4]. This method requires that the time in all the nodes is synchronized and thus it is sensitive clock drifts [4].

4.2.4 Time Of Flight

This is the time taken for the electromagnetic signal to move from the source node to the target node [4]. Assuming the speed of propagation of the electromagnetic signal is known, the distance between the source node and the target node can easily be obtained using the following equation [28].

\[ d = TOF \times c \]  \hspace{1cm} (4.5)

Due to the high speed of the electromagnetic wave, the TOF measured is extremely small (in order of nanoseconds). Thus requiring the usage of complex circuitry to measure it. Also, distance measurement using the Time of flight is very sensitive to clock drifts. Thus different ranging protocols have been proposed to mitigate the error in the time of flight [28]. These ranging protocols can be discussed as follows
**Time of Arrival**

In this protocol, the time of Flight is measured by subtracting the time in which the message left the source node from the time at which the message is received at the destination node. The time at which the message leaves the source node should be embedded in the message so that it can be used to perform the calculation of Time Of Flight.

\[ TOF = t_r - t_s \quad (4.6) \]

Where \( t_r \) is the time message arrived at the destination node, and \( t_s \) is the time the message leaves the source node.

This method requires that the time in the source node and the target node should be synchronized.

**Two Way Ranging**

This ranging protocol requires the use of at least two transceivers that exchange messages with each other. In the implementation of this method, the Source Node sends a poll message to the destination node and starts to measure the round time. The destination upon receiving the poll message sends back an acknowledgment message in which, the destination node embeds the time taken since the poll message was received to the time taken when the acknowledgment message is sent. When the poll message is received in the source node, it calculates the time taken for the signal to go on a round trip. Then to calculate the Time of Flight, the following equation is used.

\[ TOF = \frac{t_{\text{round}} - t_{\text{reply}}}{2} \quad (4.7) \]
Figure 4.3 below makes it more clear how the Two Way Ranging protocol operates.

![Two Way Ranging Protocol Diagram]

Figure 4.3: Two Way Ranging Protocol (From [4])

It should be noted that this ranging protocol does not require the synchronization of the time in the source node and the destination node [28].

But still the two way ranging protocol is prone to some errors caused by time delays in sending and receiving a message which can be in order of nanoseconds and since the speed of propagation of the signal is very high, this small-time delays can result in error in the distance measurement (in order of centimeters) [4].

**Symmetric Double-sided Two way Ranging**

This ranging protocol uses additional message exchange in the traditional Two way Ranging protocol to minimize error caused by the time delay in sending and receiving the messages in the source node [28]. The addition of another message causes the signal to have two round trip times and two reply time. And using these measured times the time of flight can be obtained using the following formulation.

\[
TOF = \frac{(t_{\text{round1}} - t_{\text{reply2}}) + (t_{\text{round2}} - t_{\text{reply1}})}{4}
\]  (4.8)

The following Figure 4.4 from [4] helps to clarify the way Symmetric Double-sided Two way Ranging protocol operates.
4.3 Trilateration

This section discusses how to obtain the position estimate of a target node given distance obtained from the ranging sensors like the UWB sensors. This section will discuss a geometrical method called Trilateration to used to calculate the position of the tag node given the distances measured by the anchors [29].

Trilateration is the procedure of estimating the position of the target Node which
is in our case Tag drone from the distance measurements obtained from Source Nodes [28]. To obtain the position \((x,y)\) of the target node we will need at least three UWB sensors. The UWB sensors are attached on the frame of the anchor drone in known positions with respect to the body frame of the quadcopter [29].

Assuming the positions of the UWB sensors are kept at positions \((x_{s,i}^b, y_{s,i}^b)\) for \(i = 1,2,3\)

The positions of the UWB sensors in the inertial coordinate frame is given by the following equation.

\[
\begin{bmatrix}
    x_{s,i}^i \\
    y_{s,i}^i \\
    1
\end{bmatrix} = \begin{bmatrix}
    C_b^i & r_{bji}^i \\
    0^T & 1
\end{bmatrix} \begin{bmatrix}
    x_{s,i}^b \\
    y_{s,i}^b \\
    1
\end{bmatrix}
\]

(4.10)

Where \(C_b^i\) is the rotational matrix from the body coordinate frame of the anchor robot to the inertial coordinate frame, \(r_{bji}^i\) is the position of the anchor expressed in the inertial coordinate frame, \(x_{s,i}^i\) and \(y_{s,i}^i\) are the \(x\) and \(y\) coordinate position of the UWB sensors expressed in the inertial coordinate frame.

We are assuming all the positions are expressed in the inertial frame so we will drop the super script \(i\) for simplicity. In the ideal case, the distance between the UWB sensor, \(i\), to the tag can be expressed as follows.

\[
f(r_t) = \sqrt{(x_t - x_{s,i})^2 + (y_t - y_{s,i})^2} = d_i
\]

(4.11)

Squaring both sides, opening the brackets and rearranging the above equation will
look like the following (in matrix form)

\[
\begin{bmatrix}
    x_{s,1} & y_{s,1} & -0.5 \\
    x_{s,2} & y_{s,2} & -0.5 \\
    x_{s,3} & y_{s,3} & -0.5
\end{bmatrix}
\begin{bmatrix}
    x_t \\
    y_t \\
    (x_t^2 + y_t^2)
\end{bmatrix}
= 
\begin{bmatrix}
    x_{s,1}^2 + y_{s,1}^2 - d_1^2 \\
    x_{s,2}^2 + y_{s,2}^2 - d_2^2 \\
    x_{s,3}^2 + y_{s,3}^2 - d_3^2
\end{bmatrix}
\] (4.12)

From the above equation (4.12) we can denote the following symbolism

\[A = \begin{bmatrix}
    x_{s,1} & y_{s,1} & -0.5 \\
    x_{s,2} & y_{s,2} & -0.5 \\
    x_{s,3} & y_{s,3} & -0.5
\end{bmatrix}\] (4.13)

And

\[b = \begin{bmatrix}
    x_{s,1}^2 + y_{s,1}^2 - d_1^2 \\
    x_{s,2}^2 + y_{s,2}^2 - d_2^2 \\
    x_{s,3}^2 + y_{s,3}^2 - d_3^2
\end{bmatrix}\] (4.14)

And

\[p = \begin{bmatrix}
    x_t \\
    y_t \\
    (x_t^2 + y_t^2)
\end{bmatrix}\] (4.15)

Thus the system can be written as

\[Ap = b\] (4.16)

Thus to obtain the position of the tag, linear equation (4.16) needs to be solved. If the matrix \(A\) is full column rank then a unique solution exists and the inverse of \(A\) exists \([35]\). Since \(A\) is a square matrix the solution can be obtained using the
The following equation.

\[ \mathbf{p} = A^{-1} \mathbf{b} \]  

(4.17)

The position of the tag drone will be the first two elements of the vector \( \mathbf{p} \). Thus if the UWB ranging sensors are ideal and have no noise, the equation (4.17) will give the exact solution of \( \mathbf{p} \) and obtain the position of the tag. But this not the case in the experimental setup. The output of the UWB is affected by noise.

Thus in a realistic model, the output of the UWB sensors assumed to have additive noise. So the equation of the measurements obtained from the UWB sensors can be written as follows.

\[ d_i = f(\mathbf{r}_t) + \mathcal{N}(0, \sigma) \]  

(4.18)

But

\[ f(\mathbf{r}_t) = \|\mathbf{r}_{s,i} - \mathbf{r}_t\| \]  

(4.19)

Where \( d_i \) is the distance measurement from UWB sensor \( i = 1, 2, 3 \), \( \mathbf{r}_{s,i} \) is the position of the anchor \( i \), \( \mathbf{r}_t \) is the position of the tag, and \( \mathcal{N}(0, \sigma) \) is Gaussian Noise with zero mean standard deviation, \( \sigma \).

Since the distance measurements are corrupted by noise, the exact solution can not be obtained but rather a solution the minimizes the error will be estimated. Thus the problem is reformulated as optimization problem \([28]\). To estimate the position of the Target Node, the following optimization problem has to be solved.
\[ \hat{p} = \arg \min_p (Ap - b) \] (4.20)

Since the objective function involves linear expression, the problem can be solved using the linear least square algorithm whose solution is can be written in closed form as follows \[35\] \[28\] \[4\].

\[ \hat{p} = (A^T A)^{-1} A^T b \] (4.21)

This above equation gives the estimate of the position of the Tag drone (Target Node). But this estimate tends to be too noisy to be used especially for controlling the quadcopter. But it can be filtered using some filtering techniques to obtain a smoother estimate of the position of the Tag drone which is the next step that we are going to explain.

### 4.4 Filtering

Filtering the estimated position using the linear least square algorithm is essential to obtain a better-refined estimate of the position. There are different techniques used to do filtering present in the literature. These methods range from the simple running averages to a more sophisticated method like the Kalman filter and its derivative called the extended Kalman filter \[4\] \[29\] \[28\]. Other filtering techniques also used in the literature are the particle filters and the Monte Carlo Filtering algorithm \[46\]. For the sake of simplicity, only the two filtering techniques are discussed and used in this thesis. The brief discussion of these filtering techniques is as following.
4.4.1 Exponential Smoothing

This technique tries to ensure the smoothness of the value by avoiding abrupt jumps between consecutive data points in a signal. In other words there this filtering technique enforces a relationship between the data points that guarantees gradual change of the data points in a time window[4]. This relationship is done using the exponential smoothing technique which can be expressed mathematically as follows.

\[
    s_k = \begin{cases} 
    x_k, & k = 0 \\
    \alpha \cdot x_k + (1 - \alpha) \cdot s_{k-1}, & k > 0 
    \end{cases}
\]  

(4.22)

Where \( s_k \) is the smoothened signal value at time \( k \), \( x_k \) is an unfiltered data, and the value of \( \alpha \) is between 0 and 1 [4].

As \( \alpha \) is close to 1 the filter does not filter out the noise and leave the signal the way it was. And as \( \alpha \) gets close 0, the filter makes the noisy signal smoother.

4.4.2 Kalman Filter

Kalman filter is a recursive Bayesian filter that is used to estimate the state’s system given noisy measurement data [28][46]. This filter consists of two steps which are prediction steps and the correction step[46].

The models used in Kalman filter are discretized linear dynamic model [28][4]. The general form of the model is shown in the equations below

\[
x_k = A_k x_{k-1} + B_k u_k + w_k
\]  

(4.23)

Where \( A_k \) is a state transition matrix, \( B_k \) is the input matrix, \( x_k \) and \( x_{k-1} \) are the state of the system at time step \( k \) and \( k - 1 \), respectively; \( w_k \sim \mathcal{N}(0, Q_k) \) is process noise which is a zero mean multi-variate normal distribution with covariance
matrix $Q_k$.

The measurement model is represented using the following equation.

$$z_k = C_k x_k + v_k$$  (4.24)

Where $z_k$ is the measured quantity which can be a linear combination of state vector, $C_k$ is a measurement matrix that maps the states $x_k$ to measured quantity, and $v_k \sim \mathcal{N}(0, R_k)$ is the measurement noise which is a zero-mean multivariate-normal distribution with covariance matrix $R_k$.

The Kalman filter consists of two steps namely the prediction and correction step.

**Prediction step**

In this step the new states are predicted using the previous knowledge of the old state and without taking into account measurements [16]. In this step, we use a state-space dynamic model which are showing the transition of the states from one-time stamp to the other.

$$\hat{x}_{k|k-1} = A_k \hat{x}_{k-1|k-1} + B_k u_k$$  (4.25)

where $\hat{x}_{k|k-1}$ is the prior estimate of the state given the previous estimate of the state of the system $\hat{x}_{k-1|k-1}$

$$P_{k|k-1} = A_k P_{k-1|k-1} A_k^T + Q_k$$  (4.26)

Where $P_{k|k-1}$ is the prior error covariance given the previous error covariance $P_{k-1|k-1}$
Correction step

In this step, we make use of the measurements to refine the estimate of the states done in the Prediction step.

Optimal Kalman gain is calculated using the following equation

$$K_k = P_{k|k-1}C_k^T(C_kP_{k|k-1}C_k^T + R_k)^{-1}$$  \hspace{1cm} (4.27)

The optimal Kalman Gain is then used to correct the prior error covariance estimate to obtain a posterior estimate of the error covariance matrix as indicated in the equation below.

$$P_{k|k} = (I - K_kC_k)P_{k|k-1}$$  \hspace{1cm} (4.28)

Also, the optimal Kalman Gain is then used to correct the prior state estimate to obtain a posterior estimate of the states as indicated in the equation below.

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(z_k - C_k\hat{x}_{k|k-1})$$  \hspace{1cm} (4.29)

4.5 Peer-to-Peer Localization between Agents in a Multi-Agent System

Now having briefly discussed background materials of some selected topics in localization, the algorithm used to do peer to peer localization between agents in a multi-agent system is discussed. This work exploits in detail two algorithms used to do peer to peer localization, compare them both in the simulation and in the real experimental setup.
The two algorithms are the Drift localization Algorithm and the Swarm localization algorithm. These two algorithms will be discussed in turn and see the comparison in their performance to do state estimation will be discussed. In this thesis, the state is taken as a vector having x and y coordinates of the position and velocity of the target agent (localized agent called the tag).

4.5.1 Drift Localization Algorithm

This localization algorithm has been introduced in different works [1] [28] [29]. In [28], this procedure was used to localize a mobile agent using stationary UWB anchors set at known positions. [28] introduced different flavors of this algorithm in which the IMU sensor measurements are fused in the model of the Kalman filter and the performance of different methods in estimating the states of the target robot are compared. [29] slightly use a different approach in which anchors are not fixed but attached in a mobile agent which is used to localize another mobile agent. Also [29] uses the Extended Kalman Filter to refine the state estimation of the target agent which could be a ground vehicle or aerial vehicle.

The underlying assumption of the Drift algorithm is that the mobile agent in our case an aerial vehicle (tag) is assumed to have a very small change in accelerations which are almost zero. This implies that the velocity is almost constant and the velocity drifts around the mean velocity and thus what inspired the name of the algorithm. This can be expressed mathematically as follows.

\[
\frac{dv_t}{dt} = a
\]  

(4.30)

Where \(a\) is the acceleration of the tag drone assumed to be very small.

In other words, the acceleration can be assumed to be having a normal distribution.
with a mean zero and standard deviation $\sigma$, $a \sim \mathcal{N}(0, \sigma)$

Writing equation (4.30) in a discretized form we will have;

\[ \hat{v}_{k+1|k} = \hat{v}_{k|k} + \mathcal{N}(0, \sigma) \]  

(4.31)

We also know the relationship between the displacement and velocity

\[ \frac{dr_t}{dt} = v \]  

(4.32)

Discretizing the above equation (4.32) will result to

\[ r_{k+1|k} = r_{k|k} + \Delta t \cdot v_{k|k} \]  

(4.33)

The states of the tag drone can be denoted as $x$ is obtained by stacking the position and velocity estimates as follows.

\[ x_{k|k} = \begin{bmatrix} r_{k|k} \\ v_{k|k} \end{bmatrix} \]  

(4.34)

The above linear equation (4.30) and (4.33) can be used as a model in the Kalman Filtering algorithm explained in the previous section as follows
Prediction Step

Now the equations (4.30) and (4.33) can be written in Matrix form as

\[
x_{k+1|k} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x_{k|k} + \begin{bmatrix} 0 \\ 0 \\ \mathcal{N}(0, \sigma) \\ \mathcal{N}(0, \sigma) \end{bmatrix}
\]  

(4.35)

Using the three UWB sensors to provide the range measurements, trilateration and the linear least algorithm can be used to provide the estimation of the position of a tag using the equation (4.21) which is rewritten here again as follows

\[
\hat{p} = (A^T A)^{-1} A^T b
\]

(4.21)

Thus the measured position of the tag are first two components of the vector \(\hat{p}\).

\[
z_k = \begin{bmatrix} \hat{p}(1) \\ \hat{p}(2) \end{bmatrix}
\]

(4.36)

The predicted Process noise covariance matrix equation will be

\[
P_{k|k-1} = A_k P_{k-1|k-1} A_k^T + Q_k
\]

(4.26)

Where \(P_{k|k}\) is a 4-by-4 matrix, for simplicity it can be initialized as constant \(\sigma_p\) times a 4-by-4 identity matrix

\[
P_{0|0} = \sigma_p \cdot I_{4\times4}
\]

(4.37)
Also, $A_k$ can be obtained from linear equation (4.35) above

$$A_k = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(4.38)

And finally, $Q_k$ is a 4-by-4 matrix which can simply be assumed to be a constant $\sigma_q$ times a 4-by-4 identity matrix

$$Q_k = \sigma_q \cdot I_{4\times 4}$$

(4.39)

Correction Step

The Kalman Gain was obtained using the following equation which was explained in the Kalman filtering algorithm.

$$K_k = P_{k|k-1}C_k^T(C_kP_{k|k-1}C_k^T + R_k)^{-1}$$

(4.27)

Where

$$C_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

(4.40)

And $R_k$ is a 2-by-2 matrix positive semi-definite that can be simply be assumed to be a constant multiplied by a 2-by-2 identity matrix.

$$R_k = \sigma_r \cdot I_{2\times 2}$$

(4.41)
The correction of the process covariance matrix using the optimal Kalman Gain is done using the following equation.

\[ P_{k|k} = (I - K_k C_k) P_{k|k-1} \] (4.28)

And the correction of the state estimate using the optimal Kalman Gain is done using the following equation

\[ \hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k (z_k - C_k \hat{x}_{k|k-1}) \] (4.29)

### 4.5.2 Simulation Results

The drift localization algorithm was implemented in the python code and uploaded to a simulation of an anchor drone in Gazebo using the ROS framework.

In that simulation, we have two drones the tag drone and the anchor drone. The anchor drone is controlled using a joystick. The tag drone is following the anchor drone using the formation control law introduced in Chapter 3. Thus, in this simulation setup, the two drones are swarming and we interested to see the estimate of states of the tag drone obtained using the drift localization algorithm.

The following graphs shows the estimations in the states of the tag using the drift localization algorithm when the filter parameters are set as follows \( \sigma_p = 1, \sigma_q = 0.001^2 \) and \( \sigma_r = 0.0005^2 \).
Figure 4.5: X position estimate using Drift Localization with $\sigma_p = 1, \sigma_q = 0.001^2$ and $\sigma_r = 0.0005^2$ in Simulation

Figure 4.6: Y position estimate using Drift Localization with $\sigma_p = 1, \sigma_q = 0.001^2$ and $\sigma_r = 0.0005^2$ in Simulation
4.5.3 Swarm Localization Algorithm

The Drift localization algorithm does not address the fact that the Tag drone is following the anchor drone in its model. When the agents are cooperating in implementing a task, most of the time they are working in such a way that they appear to be following each other or in other words moving together. This behavior is known as Swarming. Swarming is a behavior that is even exhibited in natural phenomena.
For example, when a group of ants is cooperating in carrying a certain load they do so while moving together as a group. This behavior is also exhibited by birds who are migrating from one point of the world to the other. The birds move together in a swarm where each bird is maintaining a constant relative position between each other. This behavior is also shown by a school of fish moving together in oceans in search of food.

Since the swarming behavior appears naturally in many natural and artificial multi-agent applications, it becomes necessary to incorporate it in our models which are used to predict or estimate the states of agents in multi-agent systems. Thus, inspiring a new localization algorithm called the swarm localization algorithm.

\[
\frac{dv_i}{dt} = \alpha (v_a - v_t) \quad (4.42)
\]

Descritizing the above equation \(4.42\)

\[
v_{t,k+1} - v_{t,k} = \alpha (v_{a,k} - v_{t,k}) \Delta t \quad (4.43)
\]

After doing some algebraic manipulations, and adding a random disturbance to the above equation \(4.43\) which can be modeled as having a normal distribution with mean zero and standard deviation, \(\sigma\).

\[
v_{t,k+1|k} = (1 - \alpha \cdot \Delta t) \cdot v_{t,k|k} + \alpha \cdot \Delta t \cdot v_{a,k|k} + \mathcal{N}(0,\sigma) \quad (4.44)
\]

Also, discretizing the equation \(4.32\) which gives the relationship between the displacement and velocity will result in equation \(4.33\)

\[
r_{k+1|k} = r_{k|k} + \Delta t \cdot v_{k|k} \quad (4.33)
\]
The state of the tag drone, denoted by \( x \), is obtained as follows

\[
x_{k|k} = \begin{bmatrix} r_{k|k} \\ v_{k|k} \end{bmatrix}
\]  

The above linear equations can be used as models for the Kalman filter algorithm explained in the previous section. Kalman filter algorithm consists of two steps namely the prediction and the correction steps which will be discussed briefly as follows.

**Prediction Step**

The linear equations (4.33) and (4.44) can be rewritten in matrix form as follows

\[
x_{k+1|k} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 - \alpha \cdot \Delta t & 0 \\ 0 & 0 & 0 & 1 - \alpha \cdot \Delta t \end{bmatrix} x_{k|k} + \begin{bmatrix} 0 \\ 0 \\ \alpha \cdot \Delta t \\ 0 \end{bmatrix} v_{a,k|k} + \mathcal{N}(0, \sigma)
\]

Using the three UWB sensors to provide the range measurements, trilateration and the linear least algorithm can be used to provide the estimation of the position of a tag using the equation (4.21) which is rewritten here again as follows

\[
\hat{p} = (A^T A)^{-1} A^T b
\]

Thus the measured position of the tag are first two components of the vector \( \hat{p} \).

\[
z_k = \begin{bmatrix} \hat{p}(1) \\ \hat{p}(2) \end{bmatrix}
\]
The predicted Process noise covariance matrix equation will be

\[
P_{k|k-1} = A_k P_{k-1|k-1} A_k^T + Q_k
\]  

(4.26)

Where \( P_{k|k} \) is a 4-by-4 matrix, for simplicity it can be initialized as constant \( \sigma_p \) times a 4-by-4 identity matrix

\[
P_{0|0} = \sigma_p \cdot I_{4 \times 4}
\]  

(4.46)

Also, \( A_k \) can be obtained from linear equation (4.45) above

\[
A_k = \begin{bmatrix}
1 & 0 & \Delta t & 0 \\
0 & 1 & 0 & \Delta t \\
0 & 0 & 1 - \alpha \cdot \Delta t & 0 \\
0 & 0 & 0 & 1 - \alpha \cdot \Delta t
\end{bmatrix}
\]  

(4.47)

And finally, \( Q_k \) is a 4-by-4 matrix which can simply be assumed to be a constant \( \sigma_q \) times a 4-by-4 identity matrix

\[
Q_k = \sigma_q \cdot I_{4 \times 4}
\]  

(4.48)

**Correction Step**

The Kalman Gain was obtained using the following equation which explained in the Kalman filtering algorithm.

\[
K_k = P_{k|k-1} C_k^T \left( C_k P_{k|k-1} C_k^T + R_k \right)^{-1}
\]  

(4.27)
Where

\[ C_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \] (4.49)

And \( R_k \) is a 2-by-2 matrix positive semi-definite that can be simply be assumed to be a constant multiplied by a 2-by-2 identity matrix.

\[ R_k = \sigma_r \cdot I_{2 \times 2} \] (4.50)

The correction of the process covariance matrix using the optimal Kalman gain is shown in the following equation.

\[ P_{k|k} = (I - K_k C_k) P_{k|k-1} \] (4.28)

The correction of the prior state estimate using the optimal Kalman gain to obtain the posterior state estimate is done using the following equation.

\[ \hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k (z_k - C_k \hat{x}_{k|k-1}) \] (4.29)

### 4.5.4 Simulation Results

The swarm localization algorithm was implemented in the python code and uploaded to a simulation of an anchor drone in Gazebo using the ROS framework.

In that simulation, we have two drones the tag drone and the anchor drone. The anchor drone is controlled using a joystick. The tag drone is following the anchor drone using the formation control law introduced in Chapter 3. Thus, in this simulation setup, the two drones are swarming and we are interested to see the estimate of states of the tag drone obtained using the swarm localization algorithm.
The following graphs show the estimations in the states of the tag using the swarm localization algorithm when the filter parameters are set as follows $\sigma_p = 1, \sigma_q = 0.001^2$ and $\sigma_r = 0.05^2$.

Figure 4.9: X position estimate using Swarm Localization with $\sigma_p = 1, \sigma_q = 0.001^2$ and $\sigma_r = 0.05^2$

![X position estimate graph](image)

Figure 4.10: Y position estimate using Swarm Localization with $\sigma_p = 1, \sigma_q = 0.001^2$ and $\sigma_r = 0.05^2$ in Simulation

![Y position estimate graph](image)
Figure 4.11: Vx velocity estimate using Swarm Localization with \( \sigma_p = 1, \sigma_q = 0.001^2 \) and \( \sigma_r = 0.05^2 \) in Simulation

Figure 4.12: Vy velocity estimate using Swarm Localization with \( \sigma_p = 1, \sigma_q = 0.001^2 \) and \( \sigma_r = 0.05^2 \) in Simulation

4.6 Comparison Between the Drift and the Swarm Localization using simulation results

This section discusses the comparison between the drift localization algorithm and the swarm localization algorithm in estimating the pose of the tag drone. The root-mean-square error (RMSE) is going to be used as the performance measure. A higher root mean square error indicates the estimation of the state is not good, and a lower
Table 4.2: RMSE in state estimates using Drift Localization Algorithm as $\sigma_p$ varies in Simulation

<table>
<thead>
<tr>
<th>$\sigma_p$</th>
<th>RMSE in $X$ (m)</th>
<th>RMSE in $Y$ (m)</th>
<th>RMSE in $v_x$ (m/s)</th>
<th>RMSE in $v_y$ (m/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.3722</td>
<td>0.3982</td>
<td>0.7574</td>
<td>0.8049</td>
</tr>
<tr>
<td>1</td>
<td>0.4078</td>
<td>0.2971</td>
<td>0.8106</td>
<td>0.6506</td>
</tr>
<tr>
<td>0.1</td>
<td>0.3629</td>
<td>0.4386</td>
<td>0.7612</td>
<td>0.8952</td>
</tr>
</tbody>
</table>

Table 4.3: RMSE in state estimates using Swarm Localization Algorithm as $\sigma_p$ varies in Simulation

<table>
<thead>
<tr>
<th>$\sigma_p$</th>
<th>RMSE in $X$ (m)</th>
<th>RMSE in $Y$ (m)</th>
<th>RMSE in $v_x$ (m/s)</th>
<th>RMSE in $v_y$ (m/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.0472</td>
<td>0.0431</td>
<td>0.0232</td>
<td>0.0143</td>
</tr>
<tr>
<td>1</td>
<td>0.0450</td>
<td>0.0473</td>
<td>0.0213</td>
<td>0.0296</td>
</tr>
<tr>
<td>0.1</td>
<td>0.0488</td>
<td>0.0358</td>
<td>0.0298</td>
<td>0.0232</td>
</tr>
</tbody>
</table>

root mean square error indicates the opposite.

The estimation of the states of the drone is compared to the actual position and velocity of the tag drone which are being published as a topic from the simulation environment (Gazebo).

These comparisons were done by changing the values of a scalar multiplier of the error covariance matrix, $\sigma_p$, a scalar multiplier of the process noise covariance matrix, $\sigma_q$, and a scalar multiplier of the observation noise covariance matrix, $\sigma_r$.

### 4.6.1 Effect of Changing the Error Covariance Matrix

The effect of changing a scalar multiplier the error covariance matrix, $\sigma_p$, when the scalar multipliers of the process noise covariance matrix, $\sigma_q$, and observation noise covariance matrix, $\sigma_r$, are fixed have been studied.

The Table 4.2 shows the Root mean square error (RMSE) in estimating the states of the tag drone using the Drift Localization Algorithm at different values of $\sigma_p$ when the values of $\sigma_q$ and $\sigma_r$ are fixed at $0.001^2$ and $0.05^2$ respectively.

The following Table 4.3 shows RMSE in estimating the states of the tag drone using Swarm Localization Algorithm at different values of $\sigma_p$ when the values of $\sigma_q$ and $\sigma_r$ are fixed at $0.001^2$ and $0.05^2$ respectively.
Figure 4.13: X position estimate using Drift Localization with $\sigma_p = 10, \sigma_q = 0.001^2$ and $\sigma_r = 0.05^2$ in Simulation

Figure 4.14: X position estimate using Swarm Localization with $\sigma_p = 10, \sigma_q = 0.001^2$ and $\sigma_r = 0.05^2$
Figure 4.15: Y position estimate using Drift Localization with $\sigma_p = 10, \sigma_q = 0.001^2$ and $\sigma_r = 0.05^2$ in Simulation

Figure 4.16: Y position estimate using Swarm Localization with $\sigma_p = 10, \sigma_q = 0.001^2$ and $\sigma_r = 0.05^2$ in Simulation
Figure 4.17: Vx velocity estimate using Drift Localization with $\sigma_p = 10, \sigma_q = 0.001^2$ and $\sigma_r = 0.05^2$ in Simulation

Figure 4.18: Vx velocity estimate using Swarm Localization with $\sigma_p = 10, \sigma_q = 0.001^2$ and $\sigma_r = 0.05^2$ in Simulation
Figure 4.19: Vy velocity estimate using Drift Localization with $\sigma_p = 10, \sigma_q = 0.001^2$ and $\sigma_r = 0.05^2$ in Simulation

Figure 4.20: Vy velocity estimate using Swarm Localization with $\sigma_p = 10, \sigma_q = 0.001^2$ and $\sigma_r = 0.05^2$ in Simulation

It can be observed from Tables 4.2 and 4.3, changing the values of $\sigma_p$ did not cause a significant change in the root mean square error values in estimating the positions of the tag drone using the on-board UWB anchors in the anchor drone. But it is vividly clear that the Swarm Localization Algorithm produced much better estimates of the states of the tag drone compared to the Drift Localization Algorithm since the state estimates produced by Swarm Localization Algorithm had almost 10 times smaller Root mean square Error-values compared to the estimates produced by the
Table 4.4: RMSE in state estimates using Drift Localization Algorithm as $\sigma_q$ varies in Simulation

<table>
<thead>
<tr>
<th>$\sigma_q$</th>
<th>RMSE in $X$ (m)</th>
<th>RMSE in $Y$ (m)</th>
<th>RMSE in $v_x$ (m/s)</th>
<th>RMSE in $v_y$ (m/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0001$^2$</td>
<td>1.9884</td>
<td>2.7563</td>
<td>1.1991</td>
<td>1.5467</td>
</tr>
<tr>
<td>0.001$^2$</td>
<td>0.4078</td>
<td>0.2971</td>
<td>0.8106</td>
<td>0.6506</td>
</tr>
<tr>
<td>0.01$^2$</td>
<td>0.0507</td>
<td>0.0444</td>
<td>0.6111</td>
<td>0.5312</td>
</tr>
<tr>
<td>0.1$^2$</td>
<td>0.0424</td>
<td>0.0421</td>
<td>0.5427</td>
<td>0.5095</td>
</tr>
</tbody>
</table>

Table 4.5: RMSE in state estimates using Swarm Localization Algorithm as $\sigma_q$ varies in Simulation

<table>
<thead>
<tr>
<th>$\sigma_q$</th>
<th>RMSE in $X$ (m)</th>
<th>RMSE in $Y$ (m)</th>
<th>RMSE in $v_x$ (m/s)</th>
<th>RMSE in $v_y$ (m/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0001$^2$</td>
<td>0.3156</td>
<td>0.2351</td>
<td>0.0225</td>
<td>0.0255</td>
</tr>
<tr>
<td>0.001$^2$</td>
<td>0.0450</td>
<td>0.0473</td>
<td>0.0213</td>
<td>0.0296</td>
</tr>
<tr>
<td>0.01$^2$</td>
<td>0.0168</td>
<td>0.0174</td>
<td>0.0302</td>
<td>0.0227</td>
</tr>
<tr>
<td>0.1$^2$</td>
<td>0.0422</td>
<td>0.0424</td>
<td>0.0199</td>
<td>0.0192</td>
</tr>
</tbody>
</table>

Drift Localization Algorithm.

### 4.6.2 Effect of Changing the Process Noise Covariance Matrix

The effect of changing the scalar multiplier the process noise covariance matrix, $\sigma_q$, when the scalar multipliers of the error covariance matrix, $\sigma_p$ and observation noise covariance matrix, $\sigma_r$ are fixed have been studied.

The Table 4.4 below shows the Root mean square error (RMSE) in estimating the states of the tag drone using the Drift Localization Algorithm at different values of $\sigma_q$ when the values of $\sigma_p$ and $\sigma_r$ are fixed at 1 and 0.05$^2$ respectively.

The following Table 4.5 shows RMSE in estimating the states of the tag drone using Swarm Localization Algorithm at different values of $\sigma_q$ when the values of $\sigma_p$ and $\sigma_r$ are fixed at 1 and 0.05$^2$ respectively.
Figure 4.21: X position estimate using Drift Localization with $\sigma_p = 1, \sigma_q = 0.0001^2$ and $\sigma_r = 0.05^2$ in Simulation

Figure 4.22: X position estimate using Swarm Localization with $\sigma_p = 1, \sigma_q = 0.0001^2$ and $\sigma_r = 0.05^2$ in Simulation
Figure 4.23: Y position estimate using Drift Localization with $\sigma_p = 1, \sigma_q = 0.0001^2$ and $\sigma_r = 0.05^2$ in Simulation

Figure 4.24: Y position estimate using Swarm Localization with $\sigma_p = 1, \sigma_q = 0.0001^2$ and $\sigma_r = 0.05^2$ in Simulation
Figure 4.25: Vx velocity estimate using Drift Localization with $\sigma_p = 1, \sigma_q = 0.0001^2$ and $\sigma_r = 0.05^2$ in Simulation

Figure 4.26: Vx velocity estimate using Swarm Localization with $\sigma_p = 1, \sigma_q = 0.0001^2$ and $\sigma_r = 0.05^2$ in Simulation
It can be observed from Tables 4.4 and 4.5 that, as the value of $\sigma_q$ increased the RMSE in the position estimates of Tag drone using the Drift Localization Algorithm and Swarm Location algorithm are very close to each other. This is because when the value of $\sigma_q$ is high the Kalman Filter will rely more on the measurements done by the UWB to produce the position estimate of the tag drone and thus results to having close estimates in both the Drift and the Swarm Localization Algorithms. As the value of $\sigma_q$ gets higher, the Kalman Filter will rely more on the discretized linear
Table 4.6: RMSE in state estimates using Drift Localization Algorithm as $\sigma_r$ varies in Simulation

<table>
<thead>
<tr>
<th>$\sigma_r$</th>
<th>RMSE in $X$ (m)</th>
<th>RMSE in $Y$ (m)</th>
<th>RMSE in $v_x$ (m/s)</th>
<th>RMSE in $v_y$ (m/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0005$^2$</td>
<td>0.0419</td>
<td>0.0426</td>
<td>0.5179</td>
<td>0.4137</td>
</tr>
<tr>
<td>0.005$^2$</td>
<td>0.0445</td>
<td>0.0468</td>
<td>0.5147</td>
<td>0.5560</td>
</tr>
<tr>
<td>0.05$^2$</td>
<td>0.4078</td>
<td>0.2971</td>
<td>0.8106</td>
<td>0.6506</td>
</tr>
</tbody>
</table>

dynamic models (4.35) and (4.45) to produce position estimates of the tag drone and it clear that the position estimate produced using the Swarm Localization Algorithm are always better than those produced using the Drift Localization Algorithm.

Likewise, the velocity estimates of the tag drone produced using the Swarm localization algorithm is much better compared to those obtained using the Drift Localization Algorithm. The velocity estimates using both of the algorithms gradually become worse as the $\sigma_q$ decreases in value.

In contrast to the position estimates, the velocity estimates of the Swarm and Drift algorithm do not get closer as the value of the $\sigma_q$ becomes higher. This is because the UWB range sensors provide the position measurement and not the velocity measurements. Thus even when the Kalman Filter Algorithm relies on the sensor measurements, the estimate of the velocity of the tag drone is not affected very much.

### 4.6.3 Effect of Changing the Observation Noise Covariance Matrix

The effect of changing the scalar multiplier the observation noise covariance matrix, $\sigma_r$, when the scalar multipliers of the process noise covariance matrix, $\sigma_q$, and error covariance matrix, $\sigma_p$, are fixed have been studied.

The Table 4.6 below shows the Root mean square error (RMSE) in estimating the states of the tag drone using the Drift Localization Algorithm at different values of $\sigma_r$ when the values of $\sigma_p$ and $\sigma_q$ are fixed at 1 and 0.001$^2$ respectively.

The following Table 4.7 shows RMSE in estimating the states of the tag drone
Table 4.7: RMSE in state estimates using Drift Localization Algorithm as $\sigma_r$ varies in Simulation

<table>
<thead>
<tr>
<th>$\sigma_r$</th>
<th>RMSE in $X$ (m)</th>
<th>RMSE in $Y$ (m)</th>
<th>RMSE in $v_x$ (m/s)</th>
<th>RMSE in $v_y$ (m/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0005²</td>
<td>0.0417</td>
<td>0.0422</td>
<td>0.0218</td>
<td>0.0224</td>
</tr>
<tr>
<td>0.005²</td>
<td>0.0162</td>
<td>0.0168</td>
<td>0.0204</td>
<td>0.0175</td>
</tr>
<tr>
<td>0.05²</td>
<td>0.0450</td>
<td>0.0473</td>
<td>0.0213</td>
<td>0.0296</td>
</tr>
</tbody>
</table>

using Swarm Localization Algorithm at different values of $\sigma_r$ when the values of $\sigma_p$ and $\sigma_q$ are fixed at 1 and 0.001² respectively

Figure 4.29: X position estimate using Drift Localization with $\sigma_p = 1, \sigma_q = 0.001²$ and $\sigma_r = 0.0005²$ in Simulation

Figure 4.30: X position estimate using Swarm Localization with $\sigma_p = 1, \sigma_q = 0.001²$ and $\sigma_r = 0.0005²$ in Simulation
Figure 4.31: Y position estimate using Drift Localization with $\sigma_p = 1, \sigma_q = 0.001^2$ and $\sigma_r = 0.0005^2$ in Simulation

Figure 4.32: Y position estimate using Swarm Localization with $\sigma_p = 1, \sigma_q = 0.001^2$ and $\sigma_r = 0.0005^2$ in Simulation
Figure 4.33: Vx velocity estimate using Drift Localization with $\sigma_p = 1$, $\sigma_q = 0.001^2$ and $\sigma_r = 0.0005^2$ in Simulation

Figure 4.34: Vx velocity estimate using Swarm Localization with $\sigma_p = 1$, $\sigma_q = 0.001^2$ and $\sigma_r = 0.0005^2$ in Simulation
By observing the Tables 4.6 and 4.7, it is notable that as the value of the $\sigma_r$ decreases the position estimates done by both of the algorithms are closer since they have very close RMSE values. This is because when the value of $\sigma_r$ is very small the Kalman Filtering Algorithm will rely more on the measurements done by the UWB sensors to obtain the position estimates and thus resulting in very similar estimates in both of the Algorithms.

When the value of $\sigma_r$ increases the Kalman algorithm will rely more on the dis-

Figure 4.35: Vy velocity estimate using Drift Localization with $\sigma_p = 1, \sigma_q = 0.001^2$ and $\sigma_r = 0.0005^2$ in Simulation

Figure 4.36: Vy velocity estimate using Swarm Localization with $\sigma_p = 1, \sigma_q = 0.001^2$ and $\sigma_r = 0.0005^2$ in Simulation
cretized linear models (4.35) and (4.45) to obtain the position estimates and thus resulting into gradual increase in RMSE in both algorithms. But overall the position estimates done by the swarm localization algorithm is much better compared to the estimates done by the use of the Drift Localization Algorithm.

On the other hand, the velocity estimates of the swarm and the Drift localization algorithms do not get closer when the value of $\sigma_r$ is decreasing. This is because in both of the algorithms the only measurement devices used are the UWB ranging sensors which will result in position measurements rather than velocity measurements of the tag. Thus, when the Kalman filter is relying on the measurements the velocity estimates are not affected significantly.

But it Crystal clear that the velocity estimates obtained using the Swarm Localization Algorithm are better than the velocity estimates obtained using the Drift Localization Algorithm since the Swarm localization algorithm had estimates with ten folds lower RMSE values compared to the ones obtained using the Drift Localization Algorithm.
Chapter 5

Experiments

This Chapter is dedicated to explaining the experimental procedure used to validate the simulation results obtained from the previous chapter. First, the equipment used in experiments will be briefly described. And then the experimental procedure used and also how the data was collected will be explained. And then finally the results of the experiments will be analyzed.

5.1 Apparatus used for the Experiment

This section explains briefly the Apparatus used in the Experiments.

5.1.1 Decawave

The Decawave device, DW1000, is the device that is used as the UWB radios in experiment [28][29]. This device (DW1000) is reported to have very high positioning accuracy in the Line of sight and even have pretty good accuracy even the cases where there is no direct line of sight between the source and the target node [28][47]. EVB1000 includes the DW1000 chip plus the reprogrammable microprocessor, LCD screen, USB connection, and an antenna to improve the communication [28][47]. This thesis used four EVB1000 devices in which three were attached in the anchor drone and are called the anchors and one is attached to the tag drone and is called a tag.
5.1.2 Motion Capture System

This is a system that uses multiple cameras to locate the reflective balls attached to a body of interest. The Mocap system uses some optimization method to locate the position of the body very accurately and provides very high-frequency data of the position and the orientation of the object of interest.

This research uses the motion capture system position estimates as ground truth to validate the position estimated using the Drift and the swarm localization algorithm.

To use the motion capture system, attach reflective balls on the body the drone using some reflective balls. Keep the drone on the flying arena and open the motive program (Program used for the motion capture system), create the rigid body, specify the data rate you want to broadcast the rigid body and the orientation of the axis of the mocap system. The detailed procedure of using Motion capture system can be found in [48].

5.1.3 Joystick

This is a device used to send the velocity commands in the Anchor drone which is used to localize the tag drone. This device can be also used to arm the drones and disarm the motors depending on how the nodes are programmed.

5.1.4 Control Center Computer

This is a computer used to get access to the companion computers on the drones. This computer is also used to run the nodes in the companion computer remotely. Also, this computer can be used to reconfigure the flight controller (Pixhawk) using the QGroundControl.
5.1.5 On-board Computer

This is the computer that serves as the on-board computation unit. This computer runs Ubuntu 16 and ROS Kinetic. This computer is running the nodes used for Localization and it runs nodes used for navigation control of the drones.

5.1.6 Lidar Lite Range Finder

This device is used for obtaining accurate altitude measurements and use it for altitude control of the drones.

5.1.7 Airframe

The hexacopter frame with a diameter of 50 cm was used for as the anchor drone which is used for the localization. The quadcopter frame with of diameter 35 cm was used as the Tag drone.

Figure 5.1: Anchor Drone used for the Indoor Experimental setup
5.2 Experimental Procedure

5.2.1 Calibration

This is one of the most important parts of the experiment whereby the sensors are calibrated to produce accurate readings. Since the sensor which is going to be used extensively is the UWB sensor. This part explains three main calibration considerations which should be taken very seriously to ensure accurate readings from the UWB sensors.

Calibrating the bias and the standard deviation in UWB

It has been observed by [29] that the UWB sensors exhibit a non zero bias in its measurement which generally increases with the increase in the distance between the anchor and tag.

To obtain the value of the bias and the standard deviation, two UWB sensors
are kept at a known distance apart a number of distance measurements are recorded. And the mean and standard deviation of the error between the true distance and the samples of distance measurements taken by the UWB sensors are obtained. This bias and the standard deviation is compensated in the readings obtained from the UWB sensors in the codes running the localization algorithm.

**Accounting for the height difference**

When the anchor drone and the tag drone are flying. It happens that the UWB sensors are not at the same height level due to some disturbances during the flight. When this happens the distance measurement taken by the UWB is different from the distance between the two drones in the horizontal plane. Mathematically this can be expressed as follows.

\[
d_{\text{UWB}}^2 = d_{\text{hor}}^2 + h_{\text{diff}}^2
\]  

Where \(d_{\text{UWB}}\) is the distance measured by the UWB, \(d_{\text{hor}}\) distance between the drones in the horizontal plane, and \(h_{\text{diff}}\) is the height difference.

The localization code should constantly measure the height difference between the drones and use it to compensate for the difference in the distance measured from the actual horizontal distance between the drones.

**Exact measurement of the Anchor position**

The position of the Anchor UWB sensor on the frame of the anchor drone should be accurately be determined. This position is measured with respect to the center mass of the rigid body (as defined by the Motion Capture System).
5.3 Programs used for the Experiments

The actual Experiment involves launching a ROS launch file in the anchor drone that will launch three main programs which are
• The Motion Capture system node which broadcasts the pose of the drone. And this pose of the anchor is linked with the vision topic of Mavros.

• The node responsible for the navigation of anchor drone. This node makes the anchor drone follow the velocity commands given to the anchor drone from the joystick.

• A node in which the Swarm localization Algorithm is implemented

• A node in which the Drift Localization Algorithm is implemented

When the tag drone is going off-board, the formation control node is running which makes the tag drone follow the anchor drone by executing the formation control law introduced in Chapter 3.

The joystick node runs in the central computer. This joystick node sends the velocity commands to the anchor drone.

The central computer is also used to collect the data which will be used for analysis.

5.4 Results of the Experiments

This section discusses the results obtained in the experiments where the localization algorithms were implemented in the Anchor drone and used to localize a tag drone while the two drones are swarming. Also, some comparisons between the drift localization algorithm and the swarm localization algorithm in estimating the pose of the tag drone are done. The root-mean-square error(RMSE) is going to be used as the performance measure. A higher RMSE indicates the estimation of the state is not good, and a lower root mean square error indicates the opposite.

The estimation of the states of the drone done by the two localization algorithms is compared to the actual position and velocity of the tag drone which are being published by the motion capture system and the MAVROS which fuses data from the IMU and other sensors using EKF.
Table 5.1: RMSE in state estimates using Drift Localization Algorithm as $\sigma_p$ varies in Indoor Experiments

<table>
<thead>
<tr>
<th>$\sigma_p$</th>
<th>RMSE in $X$ (m)</th>
<th>RMSE in $Y$ (m)</th>
<th>RMSE in $v_x$ (m/s)</th>
<th>RMSE in $v_y$ (m/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.1404</td>
<td>0.1762</td>
<td>0.1500</td>
<td>0.1775</td>
</tr>
<tr>
<td>1</td>
<td>0.1498</td>
<td>0.1937</td>
<td>0.1336</td>
<td>0.1743</td>
</tr>
<tr>
<td>0.1</td>
<td>0.1344</td>
<td>0.1879</td>
<td>0.1269</td>
<td>0.1863</td>
</tr>
</tbody>
</table>

Table 5.2: RMSE in state estimates using Swarm Localization Algorithm as $\sigma_p$ varies in Indoor Experiments

<table>
<thead>
<tr>
<th>$\sigma_p$</th>
<th>RMSE in $X$ (m)</th>
<th>RMSE in $Y$ (m)</th>
<th>RMSE in $v_x$ (m/s)</th>
<th>RMSE in $v_y$ (m/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.1185</td>
<td>0.1971</td>
<td>0.0583</td>
<td>0.0897</td>
</tr>
<tr>
<td>1</td>
<td>0.0938</td>
<td>0.1914</td>
<td>0.0568</td>
<td>0.0766</td>
</tr>
<tr>
<td>0.1</td>
<td>0.1163</td>
<td>0.1993</td>
<td>0.0364</td>
<td>0.0804</td>
</tr>
</tbody>
</table>

These comparisons were done by changing the values of a scalar multiplier of the error covariance matrix, $\sigma_p$, a scalar multiplier of the process noise covariance matrix, $\sigma_q$, and a scalar multiplier of the observation noise covariance matrix, $\sigma_r$.

5.4.1 Effect of Changing the Error Covariance Matrix

The effect of changing a scalar multiplier of the error covariance matrix, $\sigma_p$, when the scalar multipliers of the process noise covariance matrix, $\sigma_q$, and observation noise covariance matrix, $\sigma_r$, are fixed have been studied.

The Table 5.1 shows the Root mean square error (RMSE) in estimating the states of the tag drone using the Drift Localization Algorithm at different values of $\sigma_p$ when the values of $\sigma_q$ and $\sigma_r$ are fixed at 0.001\(^2\) and 0.05\(^2\) respectively.

The following Table 5.2 shows RMSE in estimating the states of the tag drone using Swarm Localization Algorithm at different values of $\sigma_p$ when the values of $\sigma_q$ and $\sigma_r$ are fixed at 0.001\(^2\) and 0.05\(^2\) respectively.
Figure 5.5: X position estimate using Drift and Swarm Localization with $\sigma_p = 10$, $\sigma_q = 0.001^2$, and $\sigma_r = 0.05^2$ in Indoor Experiment

Figure 5.6: X position estimate using Drift and Swarm Localization with $\sigma_p = 0.1$, $\sigma_q = 0.001^2$, and $\sigma_r = 0.05^2$ in Indoor Experiment
Figure 5.7: Y position estimate using Drift and Swarm Localization with $\sigma_p = 10$, $\sigma_q = 0.001^2$, and $\sigma_r = 0.05^2$ in Indoor Experiment

Figure 5.8: Y position estimate using Drift and Swarm Localization with $\sigma_p = 0.1$, $\sigma_q = 0.001^2$, and $\sigma_r = 0.05^2$ in Indoor Experiment
Figure 5.9: Vx velocity estimate using Drift and Swarm Localization with $\sigma_p = 10$, $\sigma_q = 0.001^2$, and $\sigma_r = 0.05^2$ in Indoor Experiment

Figure 5.10: Vx velocity estimate using Drift and Swarm Localization with $\sigma_p = 0.1$, $\sigma_q = 0.001^2$, and $\sigma_r = 0.05^2$ in Indoor Experiment
Figure 5.11: Vy velocity estimate using Drift and Swarm Localization with $\sigma_p = 10$, $\sigma_q = 0.001^2$, and $\sigma_r = 0.05^2$ in Indoor Experiment

Figure 5.12: Vy velocity estimate using Drift and Swarm Localization with $\sigma_p = 0.1$, $\sigma_q = 0.001^2$, and $\sigma_r = 0.05^2$ in Indoor Experiment

As it can be observed from the experimental results, the Tables 5.1 and 5.2 shows that changing the values of $\sigma_p$ did not cause a significant change in the root mean square error values in estimating the positions of the tag drone using the on-board UWB anchors in the anchor drone. Nevertheless, it is vividly clear that the Swarm Localization Algorithm produced much better estimates of the states of the tag drone compared to the Drift Localization Algorithm since the state estimates produced by
Table 5.3: RMSE in state estimates using Drift Localization Algorithm as $\sigma_q$ varies in Indoor Experiments

<table>
<thead>
<tr>
<th>$\sigma_q$</th>
<th>RMSE in $X$ (m)</th>
<th>RMSE in $Y$ (m)</th>
<th>RMSE in $v_x$ (m/s)</th>
<th>RMSE in $v_y$ (m/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0001$^2$</td>
<td>0.3099</td>
<td>0.4027</td>
<td>0.1902</td>
<td>0.2145</td>
</tr>
<tr>
<td>0.001$^2$</td>
<td>0.1498</td>
<td>0.1937</td>
<td>0.1336</td>
<td>0.1743</td>
</tr>
<tr>
<td>0.01$^2$</td>
<td>0.1521</td>
<td>0.1931</td>
<td>0.2084</td>
<td>0.1993</td>
</tr>
<tr>
<td>0.1$^2$</td>
<td>0.1799</td>
<td>0.2700</td>
<td>0.1919</td>
<td>0.2854</td>
</tr>
</tbody>
</table>

Table 5.4: RMSE in state estimates using Swarm Localization Algorithm as $\sigma_q$ varies in Indoor Experiments

<table>
<thead>
<tr>
<th>$\sigma_q$</th>
<th>RMSE in $X$ (m)</th>
<th>RMSE in $Y$ (m)</th>
<th>RMSE in $v_x$ (m/s)</th>
<th>RMSE in $v_y$ (m/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0001$^2$</td>
<td>0.2943</td>
<td>0.3886</td>
<td>0.0770</td>
<td>0.0823</td>
</tr>
<tr>
<td>0.001$^2$</td>
<td>0.0938</td>
<td>0.1914</td>
<td>0.0568</td>
<td>0.0766</td>
</tr>
<tr>
<td>0.01$^2$</td>
<td>0.1322</td>
<td>0.1755</td>
<td>0.0780</td>
<td>0.0811</td>
</tr>
<tr>
<td>0.1$^2$</td>
<td>0.1774</td>
<td>0.2698</td>
<td>0.0783</td>
<td>0.0811</td>
</tr>
</tbody>
</table>

Swarm Localization Algorithm had smaller Root mean square Error-values compared to the estimates produced by the Drift Localization Algorithm.

5.4.2 Effect of Changing the Process Noise Covariance Matrix

The effect of changing a scalar multiplier of the process noise covariance matrix, $\sigma_q$, when the scalar multipliers of the error covariance matrix, $\sigma_p$ and observation noise covariance matrix, $\sigma_r$ are fixed have been studied.

The Table 5.3 below shows the Root mean square error (RMSE) in estimating the states of the tag drone using the Drift Localization Algorithm at different values of $\sigma_q$ when the values of $\sigma_p$ and $\sigma_r$ are fixed at 1 and 0.05$^2$ respectively.

The following Table 5.4 shows RMSE in estimating the states of the tag drone using Swarm Localization Algorithm at different values of $\sigma_q$ when the values of $\sigma_p$, and $\sigma_r$, are fixed at 1 and 0.05$^2$ respectively.
Figure 5.13: X position estimate using Drift and Swarm Localization with $\sigma_p = 1, \sigma_q = 0.0001^2$ and $\sigma_r = 0.05^2$ in Indoor Experiment

Figure 5.14: X position estimate using Drift and Swarm Localization with $\sigma_p = 1, \sigma_q = 0.01^2$ and $\sigma_r = 0.05^2$ in Indoor Experiment
Figure 5.15: X position estimate using Drift and Swarm Localization with $\sigma_p = 1, \sigma_q = 0.1^2$ and $\sigma_r = 0.05^2$ in Indoor Experiment.

Figure 5.16: Y position estimate using Drift and Swarm Localization with $\sigma_p = 1, \sigma_q = 0.0001^2$ and $\sigma_r = 0.05^2$ in Indoor Experiment.
Figure 5.17: Y position estimate using Drift and Swarm Localization with $\sigma_p = 1, \sigma_q = 0.01^2$ and $\sigma_r = 0.05^2$ in Indoor Experiment

Figure 5.18: Y position estimate using Drift and Swarm Localization with $\sigma_p = 1, \sigma_q = 0.1^2$ and $\sigma_r = 0.05^2$ in Indoor Experiment
Figure 5.19: Vx velocity estimate using Drift and Swarm Localization with $\sigma_p = 1, \sigma_q = 0.0001^2$ and $\sigma_r = 0.05^2$ in Indoor Experiment

Figure 5.20: Vx position estimate using Drift and Swarm Localization with $\sigma_p = 1, \sigma_q = 0.01^2$ and $\sigma_r = 0.05^2$ in Indoor Experiment
Figure 5.21: Vx velocity estimate using Drift and Swarm Localization with $\sigma_p = 1, \sigma_q = 0.1^2$ and $\sigma_r = 0.05^2$ in Indoor Experiment

Figure 5.22: Vy velocity estimate using Drift and Swarm Localization with $\sigma_p = 1, \sigma_q = 0.0001^2$ and $\sigma_r = 0.05^2$ in Indoor Experiment
It can be observed from Tables 5.3 and 5.4 that, as the value of $\sigma_q$ increased the RMSE in the position estimates of Tag drone using the Drift Localization Algorithm and Swarm Location algorithm are very close to each other. This is because when the value of $\sigma_q$ is high the Kalman Filter will rely more on the measurements done by the UWB to produce the position estimate of the tag drone and thus results to having close estimates in both the Drift and the Swarm Localization Algorithms. As the value of $\sigma_q$ gets higher, the Kalman Filter will rely more on the discretized linear
dynamic models (4.35) and (4.45) to produce position estimates of the tag drone and it clear that the position estimate produced using the Swarm Localization Algorithm are always better than those produced using the Drift Localization Algorithm. Likewise, the velocity estimates of the tag drone produced using the Swarm localization algorithm is much better compared to those obtained using the Drift Localization Algorithm. The velocity estimates using both of the algorithms gradually become worse as the $\sigma_q$ decreases in value.

In contrast to the position estimates, the velocity estimates of the Swarm and Drift algorithm do not get closer as the value of the $\sigma_q$ becomes higher. This is because the UWB range sensors provide the position measurement and not the velocity measurements. Thus even when the Kalman Filter Algorithm relies on the sensor measurements, the estimate of the velocity of the tag drone is not affected very much.

### 5.4.3 Effect of Changing the Observation Noise Covariance Matrix

The effect of changing the scalar multiplier of the observation noise covariance matrix, $\sigma_r$, when the scalar multipliers of the process noise covariance matrix, $\sigma_q$, and error covariance matrix, $\sigma_p$, are fixed have been studied.

The Table 5.5 below shows the Root mean square error (RMSE) in estimating the states of the tag drone using the Drift Localization Algorithm at different values of $\sigma_r$ when the values of $\sigma_p$ and $\sigma_q$ are fixed at 1 and $0.001^2$ respectively.

The following Table 5.6 shows RMSE in estimating the states of the tag drone
Table 5.6: RMSE in state estimates using Swarm Localization Algorithm as $\sigma_r$ varies in Indoor Experiments

<table>
<thead>
<tr>
<th>$\sigma_r$</th>
<th>RMSE in $X$ (m)</th>
<th>RMSE in $Y$ (m)</th>
<th>RMSE in $v_x$ (m/s)</th>
<th>RMSE in $v_y$ (m/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0005²</td>
<td>0.2113</td>
<td>0.1947</td>
<td>0.0891</td>
<td>0.0958</td>
</tr>
<tr>
<td>0.005²</td>
<td>0.1150</td>
<td>0.1917</td>
<td>0.0835</td>
<td>0.0795</td>
</tr>
<tr>
<td>0.05²</td>
<td>0.0938</td>
<td>0.1914</td>
<td>0.0568</td>
<td>0.0766</td>
</tr>
</tbody>
</table>

using Swarm Localization Algorithm at different values of $\sigma_r$ when the values of $\sigma_p$ and $\sigma_q$ are fixed at 1 and $0.001^2$ respectively.

Figure 5.25: X position estimate using Drift and Swarm Localization with $\sigma_p = 1, \sigma_q = 0.001^2$ and $\sigma_r = 0.005^2$ in Indoor Experiment

Figure 5.26: X position estimate using Drift and Swarm Localization with $\sigma_p = 1, \sigma_q = 0.001^2$ and $\sigma_r = 0.0005^2$ in Indoor Experiment
Figure 5.27: Y position estimate using Drift and Swarm Localization with $\sigma_p = 1, \sigma_q = 0.001^2$ and $\sigma_r = 0.05^2$ in Indoor Experiment

Figure 5.28: Y position estimate using Drift and Swarm Localization with $\sigma_p = 1, \sigma_q = 0.001^2$ and $\sigma_r = 0.0005^2$ in Indoor Experiment
Figure 5.29: Vx velocity estimate using Drift and Swarm Localization with $\sigma_p = 1, \sigma_q = 0.001^2$ and $\sigma_r = 0.05^2$ in Indoor Experiment

Figure 5.30: Vx velocity estimate using Drift and Swarm Localization with $\sigma_p = 1, \sigma_q = 0.001^2$ and $\sigma_r = 0.0005^2$ in Indoor Experiment
Figure 5.31: Vy velocity estimate using Drift and Swarm Localization with $\sigma_p = 1, \sigma_q = 0.001^2$ and $\sigma_r = 0.05^2$ in Indoor Experiment

Figure 5.32: Vy velocity estimate using Drift and Swarm Localization with $\sigma_p = 1, \sigma_q = 0.001^2$ and $\sigma_r = 0.0005^2$ in Indoor Experiment

By observing the Tables 5.5 and 5.6. It is notable that as the value of the $\sigma_r$ decreases, the position estimates done by both of the algorithms are closer since they have very close RMSE values. This is because when the value of $\sigma_r$ is very small the Kalman Filtering Algorithm will rely more on the measurements done by the UWB sensors to obtain the position estimates and thus resulting in very similar estimates in both of the Algorithms.
When the value of $\sigma_r$ increases, the Kalman algorithm will rely more on the discretized linear models (4.35) and (4.45) to obtain the position estimates and thus resulting into gradual increase in RMSE in both algorithms. But overall the position estimates done by the swarm localization algorithm is much better compared to the estimates done by the use of the Drift Localization Algorithm.

On the other hand, the velocity estimates of the swarm and the Drift localization algorithms do not get closer when the value of $\sigma_r$ is decreasing. This is because in both of the algorithms the only measurement devices used are the UWB ranging sensors which will result in position measurements rather than velocity measurements of the tag. Thus, when the Kalman filter is relying on the measurements the velocity estimates are not affected significantly.

But it is Crystal clear that the velocity estimates obtained using the Swarm Localization Algorithm are better than the velocity estimates obtained using the Drift Localization Algorithm since the Swarm localization algorithm had estimates with ten folds lower RMSE values compared to the ones obtained using the Drift Localization Algorithm.

5.4.4 Observation

All in all, it is observed that, when the two drones are in the swarming mode, the Swarm Localization Algorithm had better state estimates compared to the Drift Localization Algorithm. Thus it is safe to conclude that when the agents in a multi-agent system are exhibiting the swarming behavior, the Swarm Localization Algorithm, which accounts for the swarming behavior of the agents, produces better estimates compared to the Drift Localization Algorithm, which does not take into account the swarming behavior of the agents.
Chapter 6

Obstacle Avoidance

This chapter will briefly discuss my contribution to the European Robotics League team. European Robotic League (ERL) is a framework the conducts Robotics competitions in Europe. In those robotics tournaments, a different team from all over the world can take part to accomplish a different challenging task. The ERL conducts competitions in different countries in Europe several times in the year with different Themes [49].

6.1 Objective of the ERL tournament in February, 2019

The objective of the ERL tournament conducted in February 2019 is to design autonomous or semi-autonomous flying robots and ground vehicles which can be used for the rescue missions in outdoor and indoor scenarios.

In a nutshell, the objective of the competition is to design autonomous Unmanned Aerial Vehicle (UAV) and Ground Vehicle with the following main capabilities [49].

- The UAV or Ground vehicle should be able to navigate autonomously to a set of waypoints and being able to autonomously avoid obstacles on its way.

- The UAV or Ground vehicle should be able to detect a missing worker who will be located in one of the waypoints outdoors and another missing worker who will be situated indoors.

- The UAV or Ground vehicle should be able to deliver a first aid kit to the missing worker.
• The UAV should be able to autonomously detect the color tags which are attached in different parts of the competition arena. The red tag means Danger, the green tag means safe to go and the blue tag means the place is blocked. The Drone should use this information when it wants to use an entrance in a building.

• The UAV should be able to detect to create dense MAP once it is inside the building.

Different Members of the RISC lab concentrated on accomplishing different tasks. This thesis will discuss how the RISC team was able to solve the obstacle avoidance task.

Autonomous navigation is one of the most important requirements for many autonomous systems. This system is essential for most of the outdoors and indoors navigation systems especially in the hazardous environment during the search and rescue operations. Thus autonomous vehicles are required to be able to autonomously navigate through the environment while avoiding obstacles that might appear during the way while they are moving to the desired waypoints.

6.1.1 Existing Works in Obstacle avoidance

The obstacle avoidance problem is a very old problem in the field of robotics. And due to its crucial importance, there has been a lot of literature explaining different methodologies used to avoid obstacles in an autonomous mobile vehicle.

One of the recent works in autonomous obstacle avoidance is the PX4 avoidance package done by [50] where a pathfinding algorithm which makes use of the expected values of the risk to obtain the optimal path to the desired target in the presence of obstacles using depth cameras as the sensors. This work was able to produce good results in the simulations. But when applied to real experiments, the cameras should be finely calibrated to obtain the best results. And the performance of the depth
camera deteriorates when there are very high ambient temperature and sunlight and thus this work was not giving us suitable and reliable performance.

6.1.2 Obstacle Avoidance using the RPLidar

RPLidar is a 2 Dimensional laser scanner which gives 360 degrees scan of an environment. This device is widely used especially in the ground robots due to its low cost and its high reliability. The device can be used to navigate autonomously through the environment [5].

The RPLidar works by emitting a modulated Infrared laser signal. This laser signal is reflected by the target object and the signal which returns to the RPLidar is sampled using the vision acquisition systems found in the RPLidar and the Digital Signal Processing unit determines the distance and the angle between the RPLidar from the detected object [5].

6.2 Autonomous navigation with Obstacle Avoidance

Autonomous navigation is one of the most important requirements for many autonomous systems. This system is essential for most of the outdoors and indoors navigation systems especially in the hazardous environment during the search and rescue operations. Thus autonomous vehicles are required to be able to autonomously navigate through the environment while avoiding obstacles that might appear during the way while they are moving to the desired waypoints.

6.2.1 Existing Works in Obstacle avoidance

The obstacle avoidance problem is a very old problem in the field of robotics. And due to its crucial importance, there has been a lot of literature explaining different methodologies used to avoid obstacles in an autonomous mobile vehicle.
One of the recent works in autonomous obstacle avoidance is the PX4 avoidance package done by [50] where a pathfinding algorithm which makes use of the expected values of the risk to obtain the optimal path to the desired target in the presence of obstacles using depth cameras as the sensors. This work was able to produce good results in the simulations. But when applied to real experiments, the cameras should be finely calibrated to obtain the best results. And the performance of the depth camera deteriorates when there are very high ambient temperature and sunlight and thus this work was not giving us suitable and reliable performance.

6.2.2 Obstacle Avoidance using the RPLidar

RPLidar is a 2 Dimensional laser scanner which gives 360 degrees scan of an environment. This device is widely used especially in the ground robots due to its low cost and its high reliability. The device can be used to navigate autonomously through the environment [5].

The RPLidar works by emitting a modulated Infrared laser signal. This laser signal is reflected by the target object and the signal which returns to the RPLidar is sampled using the vision acquisition systems found in the RPLidar and the Digital Signal Processing unit determines the distance and the angle between the RPLidar from the detected object [5].
Also, the RPLidar has a motor which rotates in a clockwise direction and sequentially finding the distance between RPLidar and different points in the environment surrounding the RPLidar [5]. These distance and angles are stored in arrays and sent to the companion computer using the 3.3V TTL Serial port (USB) and used for different applications including the obstacle avoidance, localization and even can be used to do Simultaneous localization and Mapping when the rpLidar is attached on a mobile robotic system.

This work will introduce an algorithm using RPLidar to perform obstacle avoidance which was developed by the RISC ERL team.

### 6.2.3 Outdoor Obstacle Avoidance Algorithm

This algorithm was used to avoid the obstacles in the outdoor scenario. This obstacle avoidance algorithm works well in setups where there are few isolated obstacles on the path of the drone.

The drone starts by dividing the readings of the RPLidar into four main regions with the smallest or the average of the distance measurements of the RPLidar representing the distance in the particular region.
Then there will be a threshold method, which labels the region as being occupied by an obstacle or not depending on the threshold value set by the programmer.

Then if there is an obstacle in front of the drone. The drone should look at the left or the right region.

If the right region is clear. The drone should move horizontally for a certain design distance. When the drone reaches the distance. If the front is clear then the drone should orient itself in a direction towards its original path And move towards the direction until it reaches the path.

If the forward direction is not clear then the drone should repeat the same obstacle avoidance sequence.

Also, this algorithm should be executed in the same way when the front direction and the right direction are blocked in which the algorithm should avoid the obstacle by moving horizontally in the left direction until the front is clear and orient itself toward the original path and move towards the original path.

The following diagram illustrates the Outdoor obstacle avoidance algorithm.

![Diagram of Outdoor Obstacle Avoidance Algorithm](image)

Figure 6.2: Outdoor Obstacle Avoidance Algorithm

This algorithm is nicely summarized using the following pseudocode.
Algorithm 1 Move Towards the Target Waypoint while avoiding Obstacles Outdoors

Require: $d$

Ensure: $WayPointIs Reached$

while $WayPointNotReached$ do

    $d \leftarrow rpLidarMeasurement$
    $d_f \leftarrow d(\text{Front})$
    $d_r \leftarrow d(\text{Right})$
    $d_l \leftarrow d(\text{Left})$

    if $d_f < \text{Threshold}$ then
        if $d_r > \text{Threshold}$ then
            while $d_f < \text{Threshold}$ do
                MoveRight(StepSize)
            end while
            DirectTowardsGoal()
            MoveForward(StepSize)
        else
            if $d_l > \text{Threshold}$ then
                while $d_f < \text{Threshold}$ do
                    MoveRight(StepSize)
                end while
                DirectTowardsGoal()
                MoveForward(StepSize)
            else
                MoveBackward(StepSize)
                MoveRight(StepSize)
            end if
        end if
    else
        MoveForward(StepSize)
    end if
end if
This Algorithm 6.2.3 will ensure the drone will avoid the obstacle as long as the obstacle is simple. Else if the obstacle is complex the algorithm will fail.

But in practice, the algorithm works well when the obstacles are isolated that is when there is sufficient distance between the obstacles. This algorithm illustrated great performance in the outdoor search and rescue operations in the competition.

### 6.2.4 Indoor Obstacle Avoidance Algorithm

The Algorithm was used for the indoor obstacle scenario. This algorithm was designed in such a way it avoids the obstacle by moving towards the direction that avoids the obstacle.

The algorithm starts by getting the array of distances from RPLidar. The algorithm finds the set of directions where there is no obstacle. Out of all the directions the algorithm chooses the direction which brings the drone closest to the goal. The drone follows the direction in certain step distance and repeats the algorithm up until the goal is reached.

The following figure 6.3 illustrates how the Indoor Obstacle Avoidance Algorithm works.

![Figure 6.3: Outdoor Obstacle Avoidance Algorithm](image)
The following pseudocode summarizes Indoor Obstacle Algorithm

**Algorithm 2 Move Towards the Target Waypoint while avoiding Obstacles Indoors**

**Require:** $d$

**Ensure:** $WayPointIsReached$

**while** $WayPointNotReached$ **do**

$d \leftarrow rpLidarMeasurement$

$d_f \leftarrow d(\text{Front})$

**if** $d_f < \text{Threshold}$ **then**

$d_c \leftarrow d(\text{ClearFromObstacle})$

$\text{bestAzimuth} \leftarrow d_c(\text{directionBringingDroneClosestToGoal})$

$\text{DroneHeading} \leftarrow \text{bestAzimuth}$

$\text{MoveForward}(\text{StepSize})$

**else**

$\text{MoveForward}(\text{StepSize})$

**end if**

**end while**

This algorithm is more robust compared to the Outdoor algorithm when the obstacles are not isolated. That is why this algorithm is more suitable for indoor applications.

The algorithm still has a problem having a memory of what decisions did not work out. But that can be solved by using a dynamic array the stores the paths that did not work.

Overall the algorithm was able to make the drones successfully navigate indoors without hitting obstacles.
Chapter 7

Concluding Remarks

7.1 Summary

This work started by introducing the concept of localization, especially in a multi-agent system. Where the localization is a great necessity for the deployment and effectiveness of such systems. Different kinds of localization technologies and algorithms have been discussed in the literature review. The GPS technology has been the backbone for the localization in most of the military and commercial applications since it is a widespread system and uses very sophisticated technologies. But the commercially available GPS technology has weak signal penetration power and cannot be used for most of the indoor applications due to occlusion.

For indoor localization, different technologies have been tested. One of the most commonly used is the camera. These systems have been widely used due to the increase in the computational power of the on-board processing devices. Different research works were discussed where robots having an on-board camera system being used to do localization and even provide the real-time dense maps of the areas of interest. But the localization using the camera has a set back of accumulating visual drift which can be corrected by the use of loop closure which also requires additional computations.

It was reported that the UWB sensors have attracted the interest of many researchers due to its great accuracy and ability to do localization even when there is no direct line of sight in case of obstruction making it a perfect candidate for the
indoors applications. But the ultra-wideband sensors can be used with the other sensors fused by the Kalman filter to obtain better estimations of the pose of the agents as indicated by some other works.

Nevertheless, all these works did not take advantage of the behavior exhibited by the agents in the multi-agent systems when they are performing a task together. Most of the time when the agents are cooperating in undertaking a particular task they do together while exhibiting swarming behavior. This swarming behavior is also exhibited in many natural systems. So taking advantage of the swarming behaviour in the localization scheme seems to be a natural way to model agents in the multi-agent systems.

Thus this work focused on first creating the swarming behavior in the multi-agent system where the drones were used as the agents. To create this swarming behavior, the drones were controlled to maintain a constant distance between themselves. Once this control strategy has been achieved, a localization algorithm can be designed which makes use of the swarming behavior.

Furthermore, the UWB sensor technology has been revisited in greater detail. Different characteristics of the UWB sensors were outlined which gives the UWB sensor technology an upper hand in the short distance localization especially indoors. This lead to the discussion of the different ranging methods used in UWB technology where we have been introduced to four main ranging methods, out of all the Time Of Flight method is the most used method due to its better accuracy and simplicity.

The discussion went further by exploiting the positioning algorithm in which the trilateration algorithm which makes use of the distances measured by the ranging sensors to obtain the position of the target node. The trilateration method was further enhanced by generalizing it for the case where the distance measurements are affected by the additive noise which was assumed to have the normal distribution and an optimization method called least square algorithm was used to obtain the best
estimate of the position of the tag node given noisy data. Different filtering techniques have been included including the exponential smoothing and the Kalman filtering algorithm. The Kalman Filtering algorithm was seen to be a recursive Bayesian filter that finds the best estimate of the states of the tag given the noisy measurements and a linear dynamic model.

Two localization algorithms were derived which are Drift Localization Algorithm and the Swarm Localization Algorithm. Both of the localization algorithms use the output of the trilateration method as the measurement. The drift localization algorithm assumes that the velocity of the tag is slowly changing with respect to the velocity of the anchor. And Swarm localization algorithm assumes that the tag drone is trying to have the same velocity as the anchor drone. These localization algorithms have been implemented in the simulations and the real experiments where it was found that when the drones are swarming, the swarming localization algorithm obtained more accurate estimates of the position and the velocity of the tag drone.

The swarm localization algorithm then can be used as the source of feedback data for the closed-loop controlled system involving multi-agents. One of the most important applications is the search and rescue applications, the collaborative lifting applications, security and surveillance application, and many other applications where the multi-agent systems will be required.

The thesis also introduces the obstacle avoidance algorithm used in a tournament organized by the European Robotics League where rpLidar was used as a sensor to detect obstacles lying in the path of the drone. In this tournament, different teams were given a task to design autonomous aerial vehicles and ground vehicles which can be used for rescue operations. The autonomous robots are supposed to have different capabilities including obstacle avoidance, detection of a missing worker, detection of color tags and even building of 3D maps. This thesis briefly sheds some light on the two obstacle avoidance algorithms used by the RISC team, where one algorithm
was more suitable for outdoors applications and the other one suitable for indoors scenarios.

7.2 Future Research Work

The work presented in this thesis can be extended in the following directions.

- Improving the performance of the UWB localization by increasing the number states estimated including the altitude and deploy the localization and test it in the outdoor scenarios

- Using the estimated states as a feedback loop of the closed-loop system which involves multi-agent systems including the collaborative lifting, search and rescue operations

- Using the estimated states in sensor application, for instance, knowing the spatial temperature measurement over a particular area.

- Improve upon the search and rescue program used in the ERL competition and use more advanced methods like Dynamic Programming to optimally find the trajectories that lead to the missing workers while avoiding obstacles

- Improving on the search and rescue program used in ERL competition by deploying multiple robots for collaboratively searching the missing workers where these robots can localize themselves with the use of the Swarm Localization Algorithm.
REFERENCES


APPENDICES

A MATLAB/Simulink Simulations

The following program shows MATLAB simulation of the position control in a drone.

Figure A.1: Drone simulation

Inside the Position Controller block the PI controller was implemented as follows.
The following program shows the formation control in Anchor-Tag drone using MATLAB [33].
Inside the position controller of the Tag drone. The formation control law is implemented [33]. This control law is implemented as follows.
The anchor drone have a position estimator in which the Anchor is estimating the position of the tag using the Swarm Localization Algorithm and Drift Localization Algorithm. Inside the Position estimator a Kalman Filter is implemented. The implementation of the Kalman filter is as follows.

```matlab
function [x_f, y_f, Vx_f, Vy_f, error_x, error_y, error_vx, error_vy] = KALMAN(X_t, Y_t, X_a, Y_a, Z_a, Vx_a, Vy_a, Phi, The, Psi)

Rib = [cos(Psi)*cos(The) cos(Psi)*sin(The)*sin(Phi)-sin(Psi)*cos(Phi) cos(Psi)*sin(The)*cos(Phi)+sin(Psi)*sin(Phi)];

   sin(Psi)*cos(The) sin(Psi)*sin(The)*sin(Phi)+cos(Psi)*cos(Phi) sin(Psi)*sin(The)*cos(Phi)-cos(Psi)*sin(Phi);
   -sin(The) cos(The)*sin(Phi)
```
\[ \cos(\Theta) \cos(\Phi) \];

\[ T = \begin{bmatrix} \text{Rib(1,:),}X_a; \text{Rib(2,:),}Y_a; \text{Rib(3,:),}Z_a; 0,0,0,1 \end{bmatrix}; \]

\[ s_1 = T \begin{bmatrix} 1,0,0,1 \end{bmatrix}'; \]

\[ s_2 = T \begin{bmatrix} 0,0,0,1 \end{bmatrix}'; \]

\[ s_3 = T \begin{bmatrix} 0,1,0,1 \end{bmatrix}'; \]

\[ d_1 = \sqrt{(s_1(1) - X_t)^2 + (s_1(2) - Y_t)^2} + 0.02 \times \text{randn}(1,1); \]

\[ d_2 = \sqrt{(s_2(1) - X_t)^2 + (s_2(2) - Y_t)^2} + 0.02 \times \text{randn}(1,1); \]

\[ d_3 = \sqrt{(s_3(1) - X_t)^2 + (s_3(2) - Y_t)^2} + 0.02 \times \text{randn}(1,1); \]

\[ \text{Alls} = \begin{bmatrix} s_1(1) s_1(2) - 0.5; s_2(1) s_2(2) - 0.5; s_3(1) s_3(2) - 0.5 \end{bmatrix}; \]

\[ b_{lls} = 0.5 \times [s_1(1)^2 + s_1(2)^2 - d_1^2; s_2(1)^2 + s_2(2)^2 - d_2^2; s_3(1)^2 + s_3(2)^2 - d_3^2]; \]

\[ z = \text{inv}(\text{Alls}' \times \text{Alls}) \times \text{Alls}' \times b_{lls}; \]

\[ \text{meas} = [z(1); z(2)]; \]

\[ z_x = z(1); \]

\[ z_y = z(2); \]

\[ \text{use_swarm_model} = \text{true}; \]

\[ \text{sd} = 0.008; \]

% Define storage for the variables that need to persist % between time periods.

persistent P x_hat A B C Q R dt err

if isempty(P)

% First time through the code so do some initialization

\[ dt = 0.04 \times 0.04 \]

\[ err = [0; 0; 0; 0]; \]
x_hat = [0;0;0;0];
P = 0.01*eye(4);%%0.01
k1=1.5;%%k2=0;

if(use_swarm_model == true)
    A = [1 0 dt 0;0 1 0 dt;0 0 (1-k1*dt) 0;0 0 0 (1-k1*dt)];
    B = [0 0;0 0;k1*dt 0;0 k1*dt];
else
    A = [1 0 dt 0;0 1 0 dt;0 0 1 0;0 0 0 1];
    B = [0 0;0 0;1 0;0 1]
end

% B = [(dt^2)/2;(dt^2)/2;dt;dt];
C = [1 0 0 0;0 1 0 0];
Q = (sd^2)*eye(4);%%0.01,0.0001
R = (0.02^2)*eye(2);
end

% Propagate the state estimate and covariance matrix:
if (use_swarm_model == true)
    u=[Vx_a;Vy_a];
    val=B*u+[0;0;sd*randn(1,1);sd*randn(1,1)];
    x_hat = A*x_hat + B*u+[0;0;sd*randn(1,1);sd*randn(1,1) ];
end
else
    u=[sd*randn(1,1);sd*randn(1,1)];
    x_hat = A*x_hat + B*u;
end

%     u=10*randn(1,1);

P = A*P*A' + Q;
% Calculate the Kalman gain
K = P*C'/(C*P*C' + R);
% Calculate the measurement residual
resid = meas - (C*x_hat);
% Update the state and error covariance estimate
x_hat = x_hat + K*resid;
P = (eye(size(K,1))-K*C)*P;

if (use_swarm_model == false)
    limit = 1.5;

    if(x_hat(3)> limit)
        x_hat(3)=limit;
    else
        if(x_hat(3)< -limit)
            x_hat(3)=-limit
        end
    end
end

if(x_hat(4)> limit)
    x_hat(3)=limit;
else
  if(x_hat(4) < -limit)
    x_hat(4) = -limit
  end
end
end