Path Following by a Quadrotor Using Virtual Target Pursuit Guidance

Abhishek Manjunath
Utah State University

Follow this and additional works at: https://digitalcommons.usu.edu/etd

Part of the Aerospace Engineering Commons, and the Mechanical Engineering Commons

Recommended Citation
Manjunath, Abhishek, "Path Following by a Quadrotor Using Virtual Target Pursuit Guidance" (2016). All Graduate Theses and Dissertations. 4990.
https://digitalcommons.usu.edu/etd/4990

This Thesis is brought to you for free and open access by the Graduate Studies at DigitalCommons@USU. It has been accepted for inclusion in All Graduate Theses and Dissertations by an authorized administrator of DigitalCommons@USU. For more information, please contact dylan.burns@usu.edu.
PATH FOLLOWING BY A QUADROTOR USING VIRTUAL TARGET PURSUIT GUIDANCE

by

Abhishek Manjunath

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

MASTER OF SCIENCE

in

Mechanical Engineering

Approved:

Rajnikant Sharma, Ph.D. Rees Fullmer, Ph.D.
Major Professor Committee Member

David Geller, Ph.D. Mark R. McLellan, Ph.D.
Committee Member Vice President for Research and
Dean of the School of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah
2016
ABSTRACT

Path Following by a Quadrotor using Virtual Target Pursuit Guidance

by

Abhishek Manjunath, Master of Science
Utah State University, 2016

Major Professor: Rajnikant Sharma, Ph.D.
Department: Mechanical and Aerospace Engineering

Quadrotors, being more agile than fixed-wing vehicles, are the ideal candidates for autonomous missions in small, compact spaces. The immense challenge to navigate such environments is fulfilled by the concept of path following. Path following is the method of tracking/tracing a fixed, pre-defined path with minimum position error while exerting the lowest possible control effort.

In this work, the missile guidance technique of pure pursuit is adopted and modified for a 3D quadrotor model to follow fixed, compact trajectories. A specialized hardware testing platform is developed to test this algorithm. The results obtained from simulation and flight tests are compared to results from another technique called differential flatness. A small part of this thesis also deals with the stability analysis of the modified 3D pure pursuit algorithm to track trajectories expending lower control effort.

(53 pages)
PUBLIC ABSTRACT

Path Following by a Quadrotor using Virtual Target Pursuit Guidance
Abhishek Manjunath

Missile guidance laws have been solely developed to follow and intercept a target thereby destroying or damaging it. In this research, the primary objective is to modify and adopt a missile guidance law to be used on a quadrotor to follow a virtual target. The target is termed ‘virtual’ as it only exists mathematically in the form of equations. The goal is to have the quadrotor successfully following the predefined path (described by the target) while maintaining a fixed distance from the virtual target. To ensure viability and assess performance, a detailed comparison with another path following law is made.
ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor, Dr. Rajnikant Sharma, for his insightful advice and patient guidance. Without his motivation and insights this work would have never been complete. Also, I would like to express my appreciation towards my committee members: Dr. Rees Fullmer and Dr. Geller, for their valuable comments and inputs which helped me tremendously in framing this work. I would like to thank various student members of RISC Lab for their constant support, encouragement, as well as making it such a pleasant and rewarding place to work. Specifically, I would like to thank Parwinder and Anusna for their constant support. I also wish to acknowledge the help provided by Ishmael, Soodeh and Sohum.

I also take this opportunity to acknowledge the conceptual ideas and help provided by Dr. Ashwini Ratnoo, Assistant Professor, Dept. of Aerospace Engineering, IISc, Bangalore. Saurabh and Vinay, members of AVL, IISc also require a special mention here for their theoretical insights on this work.

I would like to express my gratitude to Dr. Catalin Buhusi and Dr. Mona Buhusi, Department of Psychology, USU for their timely advice and support. I would also like to thank Daniel Koch and Gary Ellingson, MAGICC Lab, BYU for their technical guidance with respect to the PX4 autopilot firmware.

I have the highest regards for the MAE department and all of the staff members, for offering me this opportunity of Masters research, as well as the financial assistance towards my tuition. I am particularly grateful for all the assistance provided by Christine Spall, who has helped me immensely through numerous paper works and formatting of the dissertation.

Last but not least, special thanks to my parents and roommates Rajesh and Niranjan for their constant support, encouragement and guidance throughout my study.

Abhishek Manjunath
# CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>iii</td>
</tr>
<tr>
<td>PUBLIC ABSTRACT</td>
<td>iv</td>
</tr>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>v</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>viii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>ix</td>
</tr>
<tr>
<td>ACRONYMS</td>
<td>xi</td>
</tr>
<tr>
<td>1 INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Background</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Contribution</td>
<td>3</td>
</tr>
<tr>
<td>1.3 Thesis Organization</td>
<td>4</td>
</tr>
<tr>
<td>2 PURE PURSUIT ALGORITHM WITH VIRTUAL TARGET GUIDANCE</td>
<td>5</td>
</tr>
<tr>
<td>2.1 Overview and Trajectory generation</td>
<td>5</td>
</tr>
<tr>
<td>2.1.1 Equations of motion of the virtual target</td>
<td>6</td>
</tr>
<tr>
<td>2.1.2 Equations of motion of the quadrotor</td>
<td>7</td>
</tr>
<tr>
<td>2.2 Pure Pursuit Implementation</td>
<td>7</td>
</tr>
<tr>
<td>2.2.1 Mapping to quadrotor commands</td>
<td>9</td>
</tr>
<tr>
<td>2.2.2 Inverse Mapping</td>
<td>9</td>
</tr>
<tr>
<td>2.3 Stability Analysis</td>
<td>11</td>
</tr>
<tr>
<td>3 ROBUST, INTELLIGENT SENSING AND CONTROL-ROS-PIXHAWK TESTING PLATFORM (RISC-RPTP)</td>
<td>15</td>
</tr>
<tr>
<td>3.1 Components of the testing platform</td>
<td>15</td>
</tr>
<tr>
<td>3.2 Setup</td>
<td>16</td>
</tr>
<tr>
<td>3.2.1 Motion Capture System</td>
<td>16</td>
</tr>
<tr>
<td>3.2.2 Aerial Platform</td>
<td>16</td>
</tr>
<tr>
<td>3.2.3 Odroid XU4</td>
<td>19</td>
</tr>
<tr>
<td>3.2.4 Pixhawk and PX4 firmware</td>
<td>21</td>
</tr>
<tr>
<td>3.2.5 Ground Control Station</td>
<td>23</td>
</tr>
<tr>
<td>4 IMPLEMENTATION AND RESULTS</td>
<td>24</td>
</tr>
<tr>
<td>4.1 Simulation</td>
<td>24</td>
</tr>
<tr>
<td>4.1.1 Results</td>
<td>24</td>
</tr>
<tr>
<td>4.2 Flight Test</td>
<td>30</td>
</tr>
<tr>
<td>5 CONCLUSION AND FUTURE WORK</td>
<td>37</td>
</tr>
<tr>
<td>5.1 Future Work</td>
<td>37</td>
</tr>
</tbody>
</table>
REFERENCES ................................................................. 39

APPENDICES ................................................................. 41
  A  ROS framework ..................................................... 42
  A.1  ROS nodes: Information/data flow ............................ 42
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Processor: AR Drone v2.0 vs Pixhawk</td>
<td>18</td>
</tr>
<tr>
<td>4.1 Circle (Simulation): Root Mean Square (RMS) error values (Convergence at 6 seconds)</td>
<td>36</td>
</tr>
<tr>
<td>4.2 Inclined figure eight (Simulation): Root Mean Square (RMS) error values (Convergence at 5 seconds)</td>
<td>36</td>
</tr>
<tr>
<td>4.3 Circle (Flight test): Root Mean Square (RMS) error values (Convergence at 20 seconds)</td>
<td>36</td>
</tr>
<tr>
<td>4.4 Inclined figure eight (Flight test): Root Mean Square (RMS) error values (Convergence at 15 seconds)</td>
<td>36</td>
</tr>
</tbody>
</table>
### LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Relative geometry between virtual target moving on a pre-defined trajectory and the UAV</td>
<td>5</td>
</tr>
<tr>
<td>2.2</td>
<td>3D Geometry for Pursuit Guidance</td>
<td>8</td>
</tr>
<tr>
<td>3.1</td>
<td>Components of RISC-RPTP</td>
<td>15</td>
</tr>
<tr>
<td>3.2</td>
<td>Motion Capture setup at RISC Lab</td>
<td>17</td>
</tr>
<tr>
<td>3.3</td>
<td>Cortex visualization of the capture volume with the quadrotor template</td>
<td>17</td>
</tr>
<tr>
<td>3.4</td>
<td>Osprey IR camera from Motion Analysis Corporation</td>
<td>18</td>
</tr>
<tr>
<td>3.5</td>
<td>Custom built quadrotor platform with Odroid and Pixhawk on-board</td>
<td>19</td>
</tr>
<tr>
<td>3.6</td>
<td>Odroid XU4</td>
<td>20</td>
</tr>
<tr>
<td>3.7</td>
<td>Pixhawk-Odroid-ROS communication</td>
<td>21</td>
</tr>
<tr>
<td>3.8</td>
<td>Pixhawk autopilot by 3D Robotics</td>
<td>22</td>
</tr>
<tr>
<td>3.9</td>
<td>Pixhawk hardware-software interfacing</td>
<td>22</td>
</tr>
<tr>
<td>4.1</td>
<td>Screen grab of the SIMULINK model depicting the flow of data</td>
<td>24</td>
</tr>
<tr>
<td>4.2</td>
<td>Circle: 3D trajectory with 0.5m/s velocity, 0.8m radius, 0.27m initial separation</td>
<td>25</td>
</tr>
<tr>
<td>4.3</td>
<td>Circle: Control effort plot with 0.5m/s velocity, 0.8m radius, 0.27m initial separation</td>
<td>25</td>
</tr>
<tr>
<td>4.4</td>
<td>Circle: Velocity plot with 0.5m/s velocity, 0.8m radius, 0.27m initial separation</td>
<td>26</td>
</tr>
<tr>
<td>4.5</td>
<td>Circle: Position error plot with 0.5m/s velocity, 0.8m radius, 0.27m initial separation</td>
<td>26</td>
</tr>
<tr>
<td>4.6</td>
<td>Figure eight: 3D trajectory with 0.16m/s velocity, 0.8m in the major axis, 1.2m in the minor axis, 0.18m initial separation</td>
<td>27</td>
</tr>
<tr>
<td>4.7</td>
<td>Figure eight: Control effort plot with 0.016m/s velocity, 0.8m in the major axis, 1.2m in the minor axis, 0.18m initial separation</td>
<td>27</td>
</tr>
</tbody>
</table>
4.8 Figure eight: Velocity plot with 0.16 m/s velocity, 0.8 m in the major axis, 1.2 m in the minor axis, 0.18 m initial separation

4.9 Figure eight: 3D position error with 0.16 m/s velocity, 0.8 m in the major axis, 1.2 m in the minor axis, 0.18 m initial separation

4.10 Time lapse image of the quadrotor moving on the circular trajectory

4.11 Circle: 3D trajectory with 0.4 m/s velocity, 0.7 m radius, 0.15 m initial separation

4.12 Circle: Control effort plot with 0.4 m/s velocity, 0.7 m radius, 0.15 m initial separation

4.13 Circle: Velocity plot with 0.4 m/s velocity, 0.7 m radius, 0.15 m initial separation

4.14 Circle: Position error plot with 0.4 m/s velocity, 0.7 m radius, 0.15 m initial separation

4.15 Inclined figure eight: 3D trajectory with 0.23 m/s velocity, 0.7 m minor axis and 1.35 m major axis, 0.36 m initial separation

4.16 Inclined figure eight: Control effort plot with 0.23 m/s velocity, 0.7 m minor axis and 1.35 m major axis, 0.36 m initial separation

4.17 Inclined figure eight: Velocity plot with 0.23 m/s velocity, 0.7 m minor axis and 1.35 m major axis, 0.36 m initial separation

4.18 Inclined figure eight: Position error plot with 0.23 m/s velocity, 0.7 m minor axis and 1.35 m major axis, 0.36 m initial separation

A.1 ROS node flowchart
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RISC</td>
<td>Robust Intelligent Sensing and Control</td>
</tr>
<tr>
<td>DF</td>
<td>Differential Flatness</td>
</tr>
<tr>
<td>TS</td>
<td>Trajectory Shaping</td>
</tr>
<tr>
<td>PP</td>
<td>Pure Pursuit</td>
</tr>
<tr>
<td>PN</td>
<td>Proportional Navigation</td>
</tr>
<tr>
<td>RPTP</td>
<td>ROS-Pixhawk Testing Platform</td>
</tr>
<tr>
<td>RTOS</td>
<td>Real Time Operating System</td>
</tr>
<tr>
<td>UAV</td>
<td>Unmanned Aerial Vehicle</td>
</tr>
<tr>
<td>PWM</td>
<td>Pulse Width Modulation</td>
</tr>
<tr>
<td>ROS</td>
<td>Robot Operating System</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>LQR</td>
<td>Linear Quadratic Regulator</td>
</tr>
<tr>
<td>PID</td>
<td>Proportional, Integral and Derivative</td>
</tr>
<tr>
<td>VT</td>
<td>Virtual Target</td>
</tr>
<tr>
<td>MoCap</td>
<td>Motion Capture</td>
</tr>
<tr>
<td>Hz</td>
<td>Hertz</td>
</tr>
<tr>
<td>m/s</td>
<td>meter per second</td>
</tr>
<tr>
<td>m</td>
<td>meter</td>
</tr>
<tr>
<td>RPC</td>
<td>Remote Procedure Call</td>
</tr>
<tr>
<td>MBps</td>
<td>Megabyte per second</td>
</tr>
</tbody>
</table>
CHAPTER 1
INTRODUCTION

Fixed pitch quadrotors are aerial platforms, highly preferred for their mechanical simplicity over other types of unmanned aerial platforms that can take-off and land vertically. Unlike traditional, fixed-wing platforms, quadrotors have the ability to hover for a prolonged period of time, as well as to perform complex maneuvers like flips and pirouettes. This unique property is what makes quadrotors to be highly desirable for navigating small, tight indoor spaces which may also include plenty of obstacles [1]. Additionally, quadrotors are also being employed to perform dangerous and tedious tasks like power line inspection, mining and even agriculture [2].

In order to achieve autonomous flight in a small, constrained environment, there is a need for robust and reliable path planning and following algorithms. Path planning is the concept of deciding on how to get from present location to the desired location in minimum time and exerting the least control effort possible. Path following is the concept of following an already preset path with minimum errors in position and velocity. In this thesis, we propose to adopt and run a 3D algorithm of pure pursuit concept (which was only for 2D fixed wing aircraft [3]) developed in this research on an agile quadrotor. A major part of this thesis also deals with the setup of the testing platform that was built for the purpose of conducting flight tests on the quadrotor.

1.1 Background

Path following techniques are conceptualized based on the required resultant flight characteristics and vehicle being used. Performance results vary depending on how a path following algorithm is tuned. For example, if the goal is to achieve quick convergence to the path, greater control effort is exerted and depending on the vehicle used with actuator lag, there might be a large overshoot in position of the vehicle.
Several researchers have developed path following algorithms for quadrotors depending on the required output behavior/result [4] [5] [6]. Out of the several path following techniques, Differential Flatness (DF) based path following approach has been shown to be advantageous for aggressive path following [5] [6]. The flip side of DF is that the noise in quadrotor states may lead to large position errors and hence, larger control effort is exerted. The reason for this behavior is that the primary objective of DF concept is to minimize the error between the states of the quadrotor and the virtual target.

There are several missile guidance laws that can be effectively used for path following using a virtual target moving on the desired path [7]. These guidance algorithms are anticipatory in nature and require small control effort in comparison to PID control and DF based path following control, as the objective is to follow the path accurately and to not intercept the virtual target. Also these techniques do not require all the vehicle states as they need only the relative position and velocity and hence can be used in GPS-denied environments as well. Park et al. [9] have used the well-known proportional navigation (PN) guidance law to track a virtual target moving at a constant velocity. It was shown that the fixed-wing UAV maintains a constant distance from the virtual target which controlled the Unmanned Aerial Vehicle (UAV) to be on a curved path. In another work [10], trajectory shaping (TS) technique has been used on a ground vehicle for virtual target following on a generalized curvature path.

Pure pursuit has been demonstrated to work successfully in simulation, on ground robots and even on full scale agricultural machinery to follow a desired trajectory [11] [12] [13] [14]. Pursuit guidance law combined with virtual target concept has been used for path following of fixed wing UAVs with a good degree of success for trajectories with large radius by Medagoda et al. [3]. It was shown that a path defined using a set of waypoints can be followed by pursuing a virtual target on the path where the velocity of the virtual target is a function of the vehicle’s velocity and the distance between the virtual target and the UAV. We extend this concept by adapting it to work on a quadrotor for smaller, compact trajectories in order to assess its performance and the control effort involved to
quantitatively assess it against the concept of DF.

1.2 Contribution

The concept of Pure Pursuit used in this research has been extensively discussed in Chapter 2. Contributions made through the research work in the thesis are as listed below;

- The missile guidance concept of Pure Pursuit (PP) used by Medagoda et al. [3] for fixed-wing aircraft was modeled in 3D and adopted for a quadcopter including the concept of virtual target guidance. This technique was modified primarily for small, compact and curved trajectories.

- The pure pursuit guidance (PP) concept was simulated for a real-world quadcopter model in SIMULINK and the performance was compared with both DF and TS algorithms. This work titled ‘Application of Virtual Target based Guidance Laws to Path Following of a Quadrotor UAV’ was accepted in the International Conference on Unmanned Aircraft Systems (ICUAS) 2016 [15].

- A stability analysis was performed to mathematically prove that the vehicle converges to the desired trajectory using the PP guidance concept. This was also proved by the simulation and hardware results.

- A robust ROS\textsuperscript{1}-Pixhawk Testing Platform (RPTP) was developed in collaboration with Parwinder Mehrok, using which this algorithm was tested. This testing platform was developed in such a way that both ground and aerial vehicles can be utilized in the future for hardware implementation of any control algorithm. Personal responsibilities in developing this platform included;

  - On-board computer (Odroid XU4) setup.

  - Changes made to the PX4 Flight Stack firmware,\textsuperscript{2} running on the Pixhawk autopilot, to accept commands from the on-board computer.

\textsuperscript{1}Robot Operating System
\textsuperscript{2}The PX4 firmware was provided by the generous members of MAGICC LAB, Brigham Young University.
– Development of the communication bridge and ROS nodes between the Ground
Control Station (GCS), on-board computer and the Pixhawk autopilot.

• All of this work, including the hardware results is due for submission to the Un-
manned Systems Journal titled *Advances in Unmanned Aircraft Systems*, published by
World Scientific.

1.3 Thesis Organization

The thesis is organized in the following way.

• Chapter 2 deals with the objective, implementation and stability analysis of the PP
algorithm used in this work.

• Chapter 3 provides details about the hardware testing platform setup and includes
specifications of the components involved.

• Chapter 4 lists out the results obtained both from simulations and flight tests while
comparing them with the results obtained using DF for similar parameters.

• Chapter 5 concludes this thesis by highlighting my contribution and its significance
for all future experiments conducted at RISC Lab.
CHAPTER 2
PURE PURSUIT ALGORITHM WITH VIRTUAL TARGET GUIDANCE

In the following section, the method involved in generating the desired trajectory and the equations of motion of the quadrotor model is presented.

2.1 Overview and Trajectory generation

Fig. 2.1: Relative geometry between virtual target moving on a pre-defined trajectory and the UAV

The primary objective of the vehicle is to follow a virtual target moving along a fixed, pre-defined, compact trajectory separated by Line of Sight (LOS) distance $R$. Fig. 2.1 describes the relative motion between the vehicle and the virtual target moving on a predefined path where $V_t$ is the target velocity, $V_v$ is the vehicle velocity, $(x_t, y_t, z_t)$ are virtual
target position co-ordinates, \((x_v, y_v, z_v)\) are vehicle position co-ordinates, \(\psi_t\) is the heading angle of the target velocity vector w.r.t x-axis while \(\theta_t\) is the pitch angle of the target velocity vector w.r.t x-y plane, \(\psi_v\) is the heading angle of the vehicle velocity vector w.r.t x-axis while \(\theta_v\) is the pitch angle of the vehicle velocity vector w.r.t x-y plane, \(\psi_l\) is the LOS heading angle measured w.r.t x-axis while \(\theta_l\) is the LOS pitch angle of the target velocity vector w.r.t x-y plane, \(R\) is the LOS vector, \(V_v\) is the velocity vector of the vehicle while \(V_t\) is the velocity vector of the target, \(a_{h}^v\) and \(a_{v}^v\) are components of vehicle accelerations in horizontal and vertical plane and \(a_{cmd}\) is the commanded lateral acceleration perpendicular to \(V_v\).

2.1.1 Equations of motion of the virtual target

The following equations are utilized to generate the required trajectory using the virtual target concept.

For a curved trajectory, the tangential velocity is a product of the angular velocity and radius of the desired curved trajectory. Knowing this, the equations of motion of the virtual target are

\[
\omega = \frac{V_t}{r}, \quad (2.1)
\]

\[
x_t = r \cos \omega t, \quad (2.2)
\]

\[
y_t = r \sin \omega t, \quad (2.3)
\]

\[
z_t = n, \quad (2.4)
\]

\[
\dot{x}_t = -r \omega \sin \omega t, \quad (2.5)
\]

\[
\dot{y}_t = r \omega \cos \omega t, \quad (2.6)
\]

\[
\dot{z}_t = 0, \quad (2.7)
\]

\[
\ddot{x}_t = -r \omega^2 \cos \omega t, \quad (2.8)
\]

\[
\ddot{y}_t = -r \omega^2 \sin \omega t, \quad (2.9)
\]

\[
\ddot{z}_t = 0, labeltgtwp \quad (2.10)
\]
where radius $r$ and velocity $V_t$ is chosen to generate the desired trajectory for DF while

$V_t = V_v \frac{R^*}{R}$ for PP, $\omega$ is the angular velocity and $n$ is a constant that is chosen to set the

height of the trajectory. $R$ is the LOS distance between the target and vehicle while $R^*$ is the

minimum value of the LOS distance $R^*$ and is a user defined parameter.

2.1.2 Equations of motion of the quadrotor

We now define the equations of motion of the unmanned aerial vehicle as the follow-
ing,

\[ \dot{x}_v = V_v \cos \theta_v \cos \psi_v, \quad (2.11) \]
\[ \dot{y}_v = V_v \cos \theta_v \sin \psi_v, \quad (2.12) \]
\[ \dot{z}_v = -V_v \sin \theta_v, \quad (2.13) \]
\[ \dot{\psi}_v = \frac{a^h_v}{V_v \cos \theta_v}, \quad (2.14) \]
\[ \dot{\theta}_v = \frac{a^v_v}{V_v}, \quad (2.15) \]

where, $V_v$ is the velocity of the vehicle, $a^v_v$ is the vertical acceleration of the vehicle and $a^h_v$ is the horizontal velocity of the vehicle, $\dot{\psi}_v$ is the yaw rate of the vehicle and $\dot{\theta}_v$ is the rate of pitch of the vehicle. When the quadrotor is considered as a point mass, $\theta_v$ is the flight path angle while $\psi_v$ is the course angle.

2.2 Pure Pursuit Implementation

This section concentrates on the equations involved in setting up the PP algorithm using the quadrotor model described previously.

The Pursuit Guidance approach by Medagoda et al. [3] involves following a virtual target using the law of pure pursuit (PP) derived from the principles of missile guidance. This approach aims to maintain its heading towards the target by driving the respective errors in azimuth ($\psi$) and elevation ($\theta$) angles to zero. The quadrotor does not intercept the
target and maintains a minimum distance ($R^*$) between itself and the virtual target. This is a highly simplified approach as only two angles are needed to be commanded in order to keep the quadrotor on the correct path. The velocities of the vehicle in $x$ ($\dot{x}_v$), $y$ ($\dot{y}_v$) and $z$ ($\dot{z}_v$) directions are used to compute the azimuth ($\psi_v$) and elevation angles ($\theta_v$) represented in Fig. 2.2 which is given by,

$$\psi_v = \tan^{-1}\left(\frac{\dot{y}_v}{\dot{x}_v}\right),$$
$$\theta_v = \tan^{-1}\left(\frac{\dot{z}_v}{\sqrt{\dot{x}_v^2 + \dot{y}_v^2}}\right).$$

The commanded azimuth angle ($\psi_c$) and elevation angle ($\theta_c$) are given by,

$$\psi_c = \tan^{-1}\left(\frac{y_t - y_v}{x_t - x_v}\right),$$
$$\theta_c = \tan^{-1}\left(\frac{z_t - z_v}{R_{xy}}\right),$$

where range in $x - y$ plane is given by,

$$R_{xy} = \sqrt{(x_v - x_t)^2 + (y_v - y_t)^2}.$$
We now compute the horizontal and vertical acceleration commands which is given by,

\[ a_h^v = K_\psi (\psi_c - \psi_v) , \]
\[ a_v^v = K_\theta (\theta_c - \theta_v) . \]

### 2.2.1 Mapping to quadrotor commands

We map the acceleration components obtained from the PP algorithm in the steps above to quadrotor commands that include thrust \((T)\), roll \((\phi)\), pitch \((\theta)\) and yaw rate \((r)\) in accordance with the work from Ferrin et al. [6].

We obtain the following accelerations in 3D by differentiating equations (2.11), (2.12) and (2.13),

\[
\begin{align*}
\ddot{x}_v &= V_v c_{\theta_c} c_{\psi_v} - \dot{\theta}_v V_v s_{\theta_c} c_{\psi_v} - \dot{\psi}_v V_v c_{\theta_c} s_{\psi_v}, \\
\ddot{y}_v &= V_v c_{\theta_c} s_{\psi_v} - \dot{\theta}_v V_v s_{\theta_c} s_{\psi_v} + \dot{\psi}_v V_v c_{\theta_c} c_{\psi_v}, \\
\ddot{z}_v &= -\dot{\theta}_v V_v c_{\theta_c} - V_v s_{\theta_c}, \\
\dot{V}_v &= k_v (V_d - V_v),
\end{align*}
\]

where \(k_v\) is the proportional gain for the velocity controller, \(V_d\) is the desired speed of the quad-rotor, \(\dot{\psi}_v\) and \(\dot{\theta}_v\) can be computed from the guidance law which are represented in equations 2.15 and 2.14. The input acceleration vector is given by,

\[
u_p = \begin{bmatrix} \ddot{p}_n_v \\ \ddot{p}_e_v \\ \ddot{p}_a_v - g \end{bmatrix}.
\]

### 2.2.2 Inverse Mapping

Inverse mapping is the method used to obtain the four inputs of throttle \((T)\), roll angle \((\phi)\), pitch angle \((\theta)\) and yaw rate \((r)\) to be provided to the quadrotor. From Ferrin et al. [6] we have \(u\) and the attitude control input vector \((v)\) can be computed for inverse map-
ping. This is very useful in the comparison as the control input vector of the two different techniques can be obtained in the exact same way.

\[
v = \begin{bmatrix} T \\ \phi \\ \theta \\ r \end{bmatrix} = \begin{bmatrix} \hat{f}_p^{-1}(x,u) \\ \hat{f}_\psi^{-1}(x,u) \end{bmatrix}.
\]

To compute thrust,

\[
u_p = R(\phi)R(\theta)R(\psi) \begin{bmatrix} 0 \\ 0 \\ \frac{T}{m} \\ -1 \end{bmatrix},
\]

where \(R(\phi)R(\theta)R(\psi)\) is the rotation matrix of the quadrotor denoted by,

\[
R(\phi)R(\theta)R(\psi) = \begin{bmatrix} 
 c(\phi)c(\psi) - c(\theta)s(\phi)s(\psi) & -c(\psi)s(\phi) - c(\phi)c(\theta)s(\psi) & s(\theta)s(\psi) \\
 c(\theta)c(\psi)s(\phi) + c(\phi)s(\psi) & c(\phi)c(\theta)c(\psi) - s(\phi)s(\psi) & -c(\psi)s(\theta) \\
s(\phi)s(\theta) & c(\phi)s(\theta) & c(\theta) 
\end{bmatrix}.
\]

Normalizing the thrust, we obtain,

\[
T = m\sqrt{u_p^T u_p}.
\]

To compute the desired roll (\(\phi_d\)) and pitch angles (\(\theta_d\)), we define an array \(Z\) as,

\[
Z = \begin{bmatrix} Z_1 \\ Z_2 \\ Z_3 \end{bmatrix} = R(\psi)u_p \begin{bmatrix} \frac{m}{-T} \\ 0 \\ 0 \end{bmatrix} = R^T(\phi)R^T(\theta) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.
\]
Solving for \( \phi \) and \( \theta \) in the above equation, we have,

\[
\phi_d = \sin^{-1}(-Z_2),
\]
\[
\theta_d = \tan^{-1}\left(\frac{Z_1}{Z_3}\right).
\]

Similarly, the last input \( r_d \) is equal to \( f_{\psi}^{-1} \) can be solved as

\[
r_d = f_{\psi}^{-1} = u_\psi \cos(\theta) \cos(\phi) - \dot{\theta} \sin(\phi),
\]

where \( \dot{\theta} = q \) can be computed by differentiating \( \theta \).

### 2.3 Stability Analysis

In this section, we attempt to prove that the system is locally stable i.e., stable at the equilibrium point using the same technique used in [16]. The equilibrium achieved is characterized by vehicle settling down at a constant altitude and moving on a horizontal circular trajectory of fixed radius.

Using relative coordinate system, we have \( \dot{R}, \dot{\psi}_{\text{LOS}} \) and \( \dot{\theta}_{\text{LOS}} \) given by,

\[
\dot{R} = V_v \cos \theta_v \cos \theta_{\text{LOS}} \cos (\psi_v - \psi_{\text{LOS}}) + V_v \sin \theta_v \sin \theta_{\text{LOS}}
\]
\[
- V_l \cos \theta_l \cos \theta_{\text{LOS}} \cos (\psi_v - \psi_{\text{LOS}}) - V_l \sin \theta_l \sin \theta_{\text{LOS}}
\]

\[
\dot{\psi}_{\text{LOS}} = \frac{V_v \cos \theta_v \sin (\psi_v - \psi_{\text{LOS}}) - V_l \cos \theta_l \sin (\psi_l - \psi_{\text{LOS})}}{R \cos \theta_{\text{LOS}}}
\]

\[
\dot{\theta}_{\text{LOS}} = \frac{-V_v \cos \theta_v \sin \theta_{\text{LOS}} \cos (\psi_v - \psi_{\text{LOS}}) - V_v \sin \theta_l \cos \theta_{\text{LOS}}}{R}
\]
\[
+ \frac{-V_l \cos \theta_l \sin \theta_{\text{LOS}} \cos (\psi_l - \psi_{\text{LOS}}) - V_l \sin \theta_v \cos \theta_{\text{LOS}}}{R}
\]
\[ \dot{\psi}_t = -\frac{K(\psi_t - \psi_{LOS})}{V_t \cos \theta_t} \]
\[ \dot{\theta}_t = -\frac{K(\theta_t - \theta_{LOS})}{V_t} \]
\[ \dot{\psi}_v = \frac{K(\pi - (\psi_v - \psi_{LOS}))}{V_v \cos \theta_v} \]
\[ \dot{\theta}_v = \frac{K(-\theta_v - \theta_{LOS})}{V_v} \]

The Jacobian matrix is defined by,

\[
J = \frac{\delta f}{\delta x} = \begin{bmatrix}
    \dot{R} & \delta \psi_t & \delta \psi_v & \delta \theta_{LOS} & \delta \theta_t & \delta \theta_v \\
    \frac{V_t + V_v}{R^2} & -\frac{K}{V_t} & 0 & 0 & 0 & 0 \\
    \frac{V_t}{R^2} & 0 & -\frac{K}{V_v} & 0 & 0 & 0 \\
    0 & 0 & 0 & 0 & -\frac{V_t}{K} & -\frac{V_v}{K} \\
    0 & 0 & 0 & -\frac{K}{V_v} & 0 & \frac{K}{V_v} \\
    0 & 0 & 0 & -\frac{K}{V_v} & 0 & \frac{K}{V_v}
\end{bmatrix} \begin{bmatrix}
    R \\
    \delta \psi_t \\
    \delta \psi_v \\
    \delta \theta_{LOS} \\
    \delta \theta_t \\
    \delta \theta_v
\end{bmatrix}
\]

where,

\[
\delta \psi_t = \psi_t - \psi_{LOS} \]
\[
\delta \psi_v = \psi_v - \psi_{LOS} \]

The average values chosen for the states are
\[ R_0 = \frac{4V^2}{K\pi}, \delta \psi_{t0} = 0, \delta \psi_{v0} = \frac{\pi}{4}, \theta_{LOS0} = 0, \theta_{t0} = 0 \] and \( \theta_{v0} = 0 \).

At equilibrium, \( V_t = V_v = V \). Assuming \( a_1 = V, a_2 = \frac{2V}{R^2}, a_3 = \frac{K}{V} \) and \( a_4 = \frac{V}{K} \).
The Jacobian matrix can now be written as,

\[ J = \begin{bmatrix}
0 & -a_1 & -a_1 & 0 & 0 & 0 \\
a_2 & -a_3 & 0 & 0 & 0 & 0 \\
a_2 & 0 & -a_3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -a_4 & -a_4 \\
0 & 0 & 0 & -a_3 & -a_3 & 0 \\
0 & 0 & 0 & -a_3 & 0 & -a_3 \\
\end{bmatrix} \]

The characteristic equation is given by,

\[ \text{det}(sI - A) = 0 \]

where, \( I \) is a 6x6 identity matrix and \( A \) is the Jacobian matrix.

The determinant of the equation was calculated using Matlab and is given by,

\[ \text{det}(sI - A) = (a_3 + s)^2(s^2 + a_3s + 2a_1a_2)(s^2 + a_3s - 2a_3a_4) = 0 \]

The roots of the characteristic equation are the eigenvalues which were obtained by solving it. The eigenvalues are,

\[ \text{eigenvalues} = \begin{bmatrix}
-\frac{a_3}{2} - \frac{\sqrt{a_3^2 - 8a_3a_4}}{2} \\
-\frac{a_3}{2} + \frac{\sqrt{a_3^2 - 8a_3a_4}}{2} \\
-\frac{a_3}{2} - \frac{\sqrt{a_3^2 - 8a_1a_2}}{2} \\
-\frac{a_3}{2} + \frac{\sqrt{a_3^2 - 8a_1a_2}}{2} \\
-a_3 \\
-a_3 \\
\end{bmatrix} \]

Now to check if the eigenvalues are truly negative, we substitute the flight test values of \( V = 0.4m/s \), \( K = 0.1 \) and \( R = 0.25m \) in the above equation. The resultant eigenvalues
are,

\[
\text{eigenvalues} = 
\begin{bmatrix}
-0.125 + 3.198i \\
-0.125 - 3.198i \\
-0.125 + 0.225i \\
-0.125 - 0.225i \\
-0.25 \\
-0.25
\end{bmatrix}
\]

Since the eigenvalues are negative, this proves that the system is stable at the equilibrium point.
CHAPTER 3

ROBUST, INTELLIGENT SENSING AND CONTROL-ROS-PIXHAWK TESTING PLATFORM (RISC-RPTP)

The testing platform was developed from the ground-up in collaboration with Parwinder Mehrok. The components of the setup and personal contributions made in this regard are detailed out in this chapter.

3.1 Components of the testing platform

![Diagram of RISC-RPTP components]

Fig. 3.1: Components of RISC-RPTP

A major part of this work was the development of the testing platform that was used to validate the PP algorithm in hardware. The Robust, Intelligent Sensing and Control ROS-Pixhawk Testing Platform (RISC-RPTP) that was systematically developed, consists of the following components;

- Motion Capture System
  - 16 Infrared (IR) camera grid
  - Cortex Software by Motion Analysis Corporation
- Ground Control Station

```
- Estimation algorithm
- ROS-Master

- Quadrotor
  - Pixhawk with modified PX4 Flight Stack
  - Odroid XU4
    * MavROS
    * MavLINK
    * Control Algorithm

3.2 Setup

This section provides in-depth details about each component of the testing platform.

3.2.1 Motion Capture System

A well planned grid of 16 Osprey IR cameras have been used to cover the entire capture volume spanning 13 feet in length, 16 feet in width and 13 feet high. The volume as visualized in the Cortex software GUI is depicted in Fig. 3.3. The layout and position of each camera in the grid as shown in Fig. 3.2 as well as the mounting structure made up of 2 inch by 4 inch wooden beams was carried out by Parwinder Mehrok. Further details about the MoCap setup can be found in [17]. The vehicle to be tracked inside the volume have a unique marker template positioned on it and the 3D position data of each marker is available to the end user at \( \approx 200 \text{ Hz} \) in the form of Ethernet packets.

3.2.2 Aerial Platform

All the initial experiments conducted at the Robust, Intelligent Sensing and Control (RISC) Lab were on an AR Drone quadrotor manufactured by Parrot SA [18]. The AR Drone is a small, off-the-shelf quadrotor which connects to the user’s phone/tablet over WiFi in order to be controlled. Even though the AR Drone was capable of stable flight
Fig. 3.2: Motion Capture setup at RISC Lab

Fig. 3.3: Cortex visualization of the capture volume with the quadrotor template
using the bottom camera for optical flow, sluggish response and latency in WiFi communication was the primary motivation to explore more agile and powerful aerial platforms. Furthermore, the proprietary firmware on the AR Drone’s autopilot prevented any useful modifications to enable on-board processing.

All the above objectives were satisfied by the 32-bit ARM based Pixhawk autopilot which originally evolved from the PX4 autopilot project, ETH, Zurich [19] [20]. A fast processor in combination with open source firmware was the ideal controller to test our algorithm at the RISC Lab. Table 3.1 provides a side by side comparison of the processor specifications of the AR Drone and Pixhawk.

<table>
<thead>
<tr>
<th>Specifications</th>
<th>Ar Drone v2.0</th>
<th>Pixhawk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>32-bit ARM Cortex A8</td>
<td>32-bit ARM STM32F427 Cortex M4 and 32-bit STM32F103 failsafe co-processor</td>
</tr>
</tbody>
</table>

A small fixed pitch quadrotor was custom built out of high strength carbon fiber, for the purpose of developing a faster, more powerful aerial platform as represented in Fig.
3.5. The detailed design, selection of the power system and testing of the quadrotor was performed by Parwinder Mehrok and can be found in [17]. The new platform enabled us to run smaller, aggressive trajectories as described in Chapter 4.

3.2.3 Odroid XU4

The Beaglebone Black was the initial choice made to do all the required computations on-board in order to reduce the latency due to WiFi as experienced in the AR Drone [21]. However, due to the limited processing power of its single core 1 GHz ARM Cortex A8 processor and 512 MB of RAM, the Beaglebone Black would run hot and slow despite the fact that high efficiency copper heat sinks were installed. Additionally, the ARM versions of ROS and Linux installed caused compatibility issues with some of the python libraries that were used.

To mitigate these problems, the Odroid XU4 was the single board computer (SBC) of
choice. With a dual processor setup combining a Samsung Exynos5422 Cortex A15 (2 GHz) processor and a Cortex A7 Octa core CPU (1.4 Ghz), coupled with 2 GB of RAM. This decision was further re-enforced by the fact that the members of the Advanced Robotic Systems Engineering Laboratory (ARSENL) from the Naval Postgraduate School, Monterey, California were able to fly 50 fixed-wing UAVs simultaneously using Pixhawk coupled with an Odroid U3\textsuperscript{1} for more than 45 minutes in harsh, hot weather [22]. In other words, this experiment proved that the Odroid was capable of handling intensive flight codes without running out of processing power even in extreme conditions of heat.

**Odroid XU4 setup**

The Odroid XU4 computer was setup with the following steps;

- A lite version on Ubuntu (LUbuntu) was loaded on the Odroid XU4 using a micro SD (Secure Digital) card and the necessary swap space was set up that was equivalent to the size of the RAM (~2 GB).

- Bare bones version of the ROS-Indigo was installed and the workspace was configured to accept the GCS as the master.

\textsuperscript{1}Predecessor to the Odroid XU4
Using a UART\(^1\) (FTDI\(^2\)) converter, the telemetry port of the Pixhawk was connected to one of the USB 3.0 ports on the Odroid XU4 as in Fig. 3.7.

MavROS and MAVLink packages were setup to enable communication from the Odroid XU4 to the Pixhawk autopilot using the *mav_pix_bridge* node. The *mav_pix_bridge* node was coded in C++ in such a way that it would run at the same rate as the controller node on the Odroid XU4 so as to be in sync.

### 3.2.4 Pixhawk and PX4 firmware

The Pixhawk pictured in Fig. 3.8 is a 32-bit ARM based autopilot which runs a Real Time Operating System (RTOS) called NuttX. The PX4 middleware runs on top of the operating system and provides device drivers and a micro object request broker (uORB) for asynchronous communication between the individual tasks running on the autopilot. The middle-ware employs a set of standard interfaces to read radio inputs, to output control commands and then to control the actuators. The PX4 firmware runs on top of this middle-ware and consists of a collection of guidance, navigation and control algorithms for fixed wing, multirotor, ground vehicle and aquatic vehicles as depicted in Fig. 3.9. It also contains estimators and filters for attitude and position. The PX4 v2 firmware was provided

---

\(^1\)Universal Asynchronous Receiver Transmitter  
\(^2\)Future Technology Devices International
Fig. 3.8: Pixhawk autopilot by 3D Robotics

Fig. 3.9: Pixhawk hardware-software interfacing
to us by the members of MAGICC Lab, Brigham Young University and it was built and compiled using the ARM Tool Chain provided by the developers of the PX4 firmware [23]. The changes made to the Pixhawk and PX4 firmware are as follows,

- The Pixhawk was modified to accept control commands from the Odroid through one of the inbuilt telemetry ports. The data rate link speed was set to an optimum value of about 0.92 MBps$^3$ (921600 baud rate).

$^3$Mega Byte per second
• The PX4 firmware was modified through commands to accept values of roll, pitch, throttle and yaw rate from the Odroid XU4.

• With the safety pilot on standby, the hierarchy for the command signals had to be set. The pilot’s radio transmitter was made the master and he was in full control of the aerial vehicle at all times. By moving the radio sticks past the set deadband PWM value for roll, pitch or yaw, the autopilot would override any autonomous commands and accept the manual control values. This was very useful as the pilot was able to prevent damage to the aerial vehicle every time there was an error with the control commands being sent from the on-board computer. This also eliminated the need for a dedicated radio switch to be assigned for off-board control mode.

• To further enhance safety during flight tests, the PX4 firmware was modified in such a way as to only accept the lowest throttle value being commanded. For example, if the Odroid was to command 70 percent throttle and the pilot had the throttle stick at 50 percent, then the accepted throttle value would be the lowest of the two i.e., the pilot’s throttle value.

3.2.5 Ground Control Station

The Ground Control Station (GCS) is a computer running Ubuntu 14.04 and ROS-Indigo on board. This computer receives the 3D marker position data over Ethernet from the Cortex Software at about 200 Hz. The individual marker data is used to estimate the vehicle position, velocity, attitude and angular rates using the estimation package on the GCS computer. These 12 states of the vehicle are sent to the on-board computer which uses them as feedback for the control loop running on it. The GCS computer is also used to control the on-board Odroid XU4 and to make changes to the control algorithm or flight parameters instantly over SSH (Secure Shell).

\footnote{Pulse Width Modulation}
CHAPTER 4
IMPLEMENTATION AND RESULTS

This chapter deals with how the algorithm was implemented both in simulation and on hardware and details out the observations from the results obtained. The first section deals with the implementation of the algorithm on SIMULINK.

4.1 Simulation

The quadrotor was modeled using the actual mass of 0.786 kg and real inertial data in SIMULINK and the PP algorithm was implemented as depicted in the figure below. For a realistic comparison so that the calculations were done at the same time-step, both the algorithms of PP and DF were run from the same SIMULINK file. This approach also made it easier to collect all the required data for plotting.

Fig. 4.1: Screen grab of the SIMULINK model depicting the flow of data

4.1.1 Results

A flat circular trajectory with radius of 0.8m and a figure eight trajectory with 1.2m in the major axis and 0.8m as the minor axis had been chosen as the reference trajectory.
Fig. 4.2: Circle: 3D trajectory with 0.5m/s velocity, 0.8m radius, 0.27m initial separation

Fig. 4.3: Circle: Control effort plot with 0.5m/s velocity, 0.8m radius, 0.27m initial separation
Fig. 4.4: Circle: Velocity plot with 0.5m/s velocity, 0.8m radius, 0.27m initial separation

Fig. 4.5: Circle: Position error plot with 0.5m/s velocity, 0.8m radius, 0.27m initial separation
Fig. 4.6: Figure eight: 3D trajectory with 0.16m/s velocity, 0.8m in the major axis, 1.2m in the minor axis, 0.18m initial separation

Fig. 4.7: Figure eight: Control effort plot with 0.16m/s velocity, 0.8m in the major axis, 1.2m in the minor axis, 0.18m initial separation
Fig. 4.8: Figure eight: Velocity plot with 0.16m/s velocity, 0.8m in the major axis, 1.2m in the minor axis, 0.18m initial separation

Fig. 4.9: Figure eight: 3D position error with 0.16m/s velocity, 0.8m in the major axis, 1.2m in the minor axis, 0.18m initial separation
A constant velocity of 0.5 m/s was chosen which provides the required angular velocity depending on the radius of the circle and a velocity of 0.16 m/s for the figure eight trajectory had been chosen. Using sampling time, angular velocity and equations of motion of the virtual target, the required waypoints for the circular and figure eight trajectories were generated. The convergence time of the PP technique depends greatly on the initial conditions of position and velocity. It was observed that the initial position of vehicle should always be behind the starting point of the virtual target with a LOS separation greater than $R^*$. It is important to note that the selected value of $R^*$ should never exceed 1.5 times the radius of the circle or the minor axis of the figure eight, as the algorithm is sensitive to initial conditions. Also, the initial velocity vector should always point towards the initial target position in order to get the vehicle to converge faster. The concept of Differential Flatness does not necessitate such kind of initial conditions for quicker convergence.

From Fig. 4.2 and Fig. 4.6, it is clear that both PP and DF algorithms perform well for both the trajectories and the vehicle stays on the trajectory. The assumption made here is that there is no noise in any of the states being used to compute the control commands. Fig. 4.3 and Fig. 4.7 represents the control inputs to the quadrotor generated from the respective guidance algorithms where it is seen that for DF, the initial roll and pitch angles are higher as the DF algorithm tends to intercept the target in the quickest possible time. Fig. 4.5 and Fig. 4.9 represents the errors in position of the quadrotor with respect to the commanded trajectory position. For the circular trajectory, the error for PP is a lot less than the error obtained for the DF approach. For the figure eight trajectory, the DF algorithm produces an overshoot that is a little more than the overshoots produced by PP technique. Fig. 4.4 and Fig. 4.8 depicts the velocities attained in order to follow the generated trajectory. It is observed that DF has a high initial velocity with the others being almost the same for the circular trajectory while there is a substantial difference in the commanded and velocity attained by the DF algorithm while following the figure eight trajectory as it seems to attain a maximum value of about 0.3 m/s from Fig. 4.8. Since only a proportional gain is used for velocity control, this phenomenon is observed. The PP
algorithm almost remains stable at the commanded velocity with 0.02 m/s magnitude.

In the next section, the hardware results were obtained using the RISC-RPTP and the resultant flight data was bagged in real-time. We also discuss the observations from the plots obtained.

4.2 Flight Test

In this section, we describe how the flight tests were carried out, the difficulties we faced and deduce our observations from the results obtained. The algorithm of PP was tested on a custom 300 mm quadrotor with a Pixhawk on board. The controls node was run from the on-board computer at 100 Hz. The controls node would be launched only after the vehicle was armed manually using the radio failing which the Pixhawk would display an error message of way-point timing out. The challenges we experienced while running our algorithm on hardware was that,

- Since the quadrotor was powered by Lithium-Polymer batteries, voltage sag after the battery reached a voltage below 11.8v, greatly affected throttle behavior during autonomous flights.

- Coupled with voltage sag was the fact that the control in z-direction was very sluggish. This greatly affected the accuracy of the data being recorded.

- The MoCap software node would shut down erratically leading to unwanted vehicle behavior eventually leading to damage to the quadrotor. To prevent this from occurring again, the accelerations were limited to a preset value in the code and the volume was re-calibrated after every 4 hours of flight.

- The Pixhawk accelerometer would develop offset/drift after running flight tests for an extended period of time leading to a bias when following the trajectory. Fortunately, this was a one-off incident and was rectified by swapping it with a newer board.
Since the virtual target depended on the range being sent using the vehicle states, the trajectory data was riddled with noise. This was countered by using a low-pass filter (LPF) on the vehicle states being used to compute range. Also the virtual target was bounded to not produce any negative velocities that were being caused by noisy vehicle data.

A circular trajectory of 0.7 m radius and an inclined figure eight trajectory with 0.7 m in the minor axis and 1.35 m in the major axis, was chosen to replicate flying in a compact environment. The velocity of the vehicle was chosen to be 0.4 m/s and 0.23 m/s for the circular and figure eight trajectory respectively while the initial separation between the vehicle and the virtual target was set to be 0.15 m for the circular trajectory and 0.36 m for the figure eight trajectory. From successive experiments, it was observed that the pure pursuit algorithm performed well when the initial position of the vehicle was set to be outside the required trajectory apart from the other mandatory requirements of initial conditions as listed in the simulation section.

Fig. 4.10: Time lapse image of the quadrotor moving on the circular trajectory
From Fig. 4.11 and Fig. 4.15, it is observed that the vehicle does converge to the trajectory using technique of PP. The vehicle takes some time to converge on to the trajectory unlike the technique of DF which intercepts the target in the minimum possible time. This re-enforces the observations made from the plots Fig. 4.5 and Fig. 4.9 obtained
Fig. 4.13: Circle: Velocity plot with 0.4 m/s velocity, 0.7 m radius, 0.15 m initial separation from simulations.

Fig. 4.14: Circle: Position error plot with 0.4 m/s velocity, 0.7 m radius, 0.15 m initial separation from simulations. Fig. 4.12 and Fig. 4.16 represents the control inputs to the quadrotor from their respective techniques. Surprisingly, the quadrotor is observed to exert greater control effort in comparison to DF algorithm. However, the initial overshoot from DF can still be observed quite distinctively in both the figures. This initial overshoot using DF can
Fig. 4.15: Inclined figure eight: 3D trajectory with 0.23 m/s velocity, 0.7 m minor axis and 1.35 m major axis, 0.36 m initial separation

Fig. 4.16: Inclined figure eight: Control effort plot with 0.23 m/s velocity, 0.7 m minor axis and 1.35 m major axis, 0.36 m initial separation

again be seen in Fig. 4.13 and Fig. 4.17 where the magnitude is as high as 1 m/s while the commanded being 0.4 m/s. This aggressive behavior is noticeable when the trajectory switches from a way-point to a circle. Fig. 4.14 and Fig. 4.18 paints a clear picture about the
Fig. 4.17: Inclined figure eight: Velocity plot with 0.23 m/s velocity, 0.7 m minor axis and 1.35 m major axis, 0.36 m initial separation

Fig. 4.18: Inclined figure eight: Position error plot with 0.23 m/s velocity, 0.7 m minor axis and 1.35 m major axis, 0.36 m initial separation

position errors obtained from both the algorithms. For such a compact trajectory of radius 0.7 m, the PP approach does perform better for the chosen set of conditions discounting the magnitude of error observed at about 14 seconds due to the switch from way-point to
trajectory when the PP technique drives the quadrotor into tail-chase mode.

From the results obtained from both simulations and flight test data, it is distinct that the concept of PP produces lower RMS values of position errors, except for the inclined circle trajectory in hardware, as depicted in Table 4.1, 4.2, 4.3 and 4.4.

Table 4.1: Circle (Simulation): Root Mean Square (RMS) error values (Convergence at 6 seconds)

<table>
<thead>
<tr>
<th>Concept</th>
<th>Complete Data</th>
<th>Convergence Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>PP</td>
<td>0.0299 m</td>
<td>0.0134 m</td>
</tr>
<tr>
<td>DF</td>
<td>0.0378 m</td>
<td>0.0197 m</td>
</tr>
</tbody>
</table>

Table 4.2: Inclined figure eight (Simulation): Root Mean Square (RMS) error values (Convergence at 5 seconds)

<table>
<thead>
<tr>
<th>Concept</th>
<th>Complete Data</th>
<th>Convergence Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>PP</td>
<td>0.1427 m</td>
<td>0.0160 m</td>
</tr>
<tr>
<td>DF</td>
<td>0.0288 m</td>
<td>0.0293 m</td>
</tr>
</tbody>
</table>

Table 4.3: Circle (Flight test): Root Mean Square (RMS) error values (Convergence at 20 seconds)

<table>
<thead>
<tr>
<th>Concept</th>
<th>Complete Data</th>
<th>Convergence Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>PP</td>
<td>0.1173 m</td>
<td>0.1078 m</td>
</tr>
<tr>
<td>DF</td>
<td>0.1446 m</td>
<td>0.1095 m</td>
</tr>
</tbody>
</table>

Table 4.4: Inclined figure eight (Flight test): Root Mean Square (RMS) error values (Convergence at 15 seconds)

<table>
<thead>
<tr>
<th>Concept</th>
<th>Complete Data</th>
<th>Convergence Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>PP</td>
<td>0.1154 m</td>
<td>0.0870 m</td>
</tr>
<tr>
<td>DF</td>
<td>0.0754 m</td>
<td>0.0752 m</td>
</tr>
</tbody>
</table>
CHAPTER 5
CONCLUSION AND FUTURE WORK

This research explored the possibility of using the *pure pursuit* missile guidance law on a quadrotor to follow a virtual target moving on a desired compact trajectory. Through this work the following objectives were fulfilled,

- The modified 3D algorithm of PP was successfully simulated using a real world quadrotor model and demonstrated the ability to follow compact paths while exerting minimum control effort (in comparison to DF). This was also proved mathematically by the stability analysis in Chapter 2.

- To validate the results obtained from simulation, the algorithm was successfully implemented on hardware.

- The systematic development of the RISC-RPTP was instrumental in realizing the PP algorithm in hardware involving the lowest possible latency in communication. Additionally, the testing platform was developed in such a way that any control algorithm including PP can be set up and tested with minimal time. Also, the same platform can be utilized to test fixed wing aerial vehicles as well as ground robots.

- The results obtained both from simulation and flight tests clearly show that the concept of *pure pursuit* does a good job of following the trajectory better than the DF concept for all cases except the figure eight trajectory. The position error for DF is marginally higher than the errors obtained for the PP concept. The PP technique was found to be highly sensitive to the initial condition parameters as well the user defined values of vehicle velocity and initial line of sight separation.

5.1 Future Work

Work in the near future could involve,
• Improving the reliability of the PP algorithm to perform better for a wider range of conditions.

• Testing the algorithm on variable pitch aerial platforms to assess changes in behavior and performance.

• Extensively testing the RPTP using GPS feedback at slower rates instead of the current MoCap feedback (200 Hz).

• Adapting the changes made to PX4 v2 firmware to newer releases in order to avail full functionality of the hardware package on Pixhawk.
REFERENCES


Appendix A
ROS framework

A.1 ROS nodes: Information/data flow

The Robot Operating System (ROS) framework for the ROS-Pixhawk Testing Platform (RPTP) is composed of several nodes that communicate with each other through streaming topics, Remote Procedure Call (RPC) services and the Parameter Server. A node is a process that performs computation. The use of nodes in ROS provides several benefits to the overall system. There is additional fault tolerance as crashes are isolated to individual nodes. Code complexity is reduced in comparison to monolithic systems. ROS supports nodes developed both in C++ and Python environments.

Fig. A.1 represents the nodes through which information/data flows using the ROS network. The /stream_markers topic publishes the 3D individual marker data which is then subscribed by the estimation package to publish the vehicle states in the /cortex_raw topic. This topic is then subscribed by both the controller and trajectory generation package on the Odroid XU4 which is used to generate the required trajectory and control commands that are sent to the Pixhawk using the mav_pix_bridge node.