QUADROTOR UAV PATH FOLLOWING USING TRAJECTORY SHAPING

by

Parwinder Singh Mehrok

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

of

MASTER OF SCIENCE

in

Mechanical Engineering

Approved:

Rajnikant Sharma, Ph.D.
Major Professor

Rees Fullmer, Ph.D.
Committee Member

David Geller, Ph.D.
Committee Member

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

UTAH STATE UNIVERSITY
Logan, Utah
2016
ABSTRACT

Quadrotor UAV Path following using Trajectory Shaping

by

Parwinder Singh Mehrok, Master of Science

Utah State University, 2016

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

This research proposes to adapt the concept of trajectory shaping guidance law on a quadrotor to follow several different predefined, compact paths. Keeping the real world nonlinearities, sensor and process noise in mind, the technique is simulated using MATLAB-Simulink. This guidance concept is implemented on a real, custom built quadrotor for which a new testing platform was developed from the ground up. Also, the results obtained from the trajectory shaping concept are compared to the performance characteristics of another path following concept called differential flatness. In addition to this, root mean square of position error data is used to compare the vehicle convergence to the desired trajectory.

(69 pages)
PUBLIC ABSTRACT

Quadrotor UAV Path following using Trajectory Shaping
Parwinder Singh Mehrok

The propose of this research is to develop a path following control law known as *Trajectory Shaping*. The approach is to develop a three-dimensional (3D) version of it from an existing two-dimensional (2D) model. Apart from research on this 3D control law, a multirotor unmanned aerial vehicle (UAV) platform, having four rotors and known as quadrotor is designed from the ground up for validating the concept of path following. Several pre-defined paths like the circle and figure-eight trajectories are used to apply the newly developed version of this control law. The existing 2D concept of *Trajectory Shaping* guidance law has already been designed and tested on a virtual point-mass vehicle model to follow predefined, compact 2D trajectories. The results obtained from new 3D guidance law are compared with another type of 3D guidance law to analyze the trajectory convergence characteristics.
To my parents, without whom I wouldn’t have achieved anything successfully
ACKNOWLEDGMENTS

I would like to express my gratitude to Dr. Rajnikant Sharma for the support, funding and guidance he has provided for my research. I am so thankful to him for his patience, motivation, and enthusiasm. Besides my advisor, I would like to thank my committee members, Dr. Rees Fullmer and Dr. David Geller for their help, guidance, insightful suggestions, and encouragement. I would also like to thank Christine Spall for her help in all the necessary paperwork and format my thesis.

I am grateful to Abhishek, who worked with me on other research projects and also very thankful to Anusna, Ishmaal, and Soodeh, the members of the RISC Lab at USU, who helped me with my research and RISC MoCap RPTP System.

Last but not the least, I would like to thank my family for their incredible support and help to achieve all my dreams in my life.

Parwinder Singh Mehrok
# 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>vi</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>ix</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>x</td>
</tr>
<tr>
<td>ACRONYMS</td>
<td>xiii</td>
</tr>
<tr>
<td>1 INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Related Work</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Research Contribution</td>
<td>2</td>
</tr>
<tr>
<td>2 TRAJECTORY SHAPING GUIDANCE</td>
<td>4</td>
</tr>
<tr>
<td>2.1 Trajectory Shaping 2D Model</td>
<td>4</td>
</tr>
<tr>
<td>2.2 3D TS Model and Mapping To Quadrotor Commands</td>
<td>6</td>
</tr>
<tr>
<td>2.3 Quadrotor Model</td>
<td>10</td>
</tr>
<tr>
<td>2.3.1 Inverse Mapping</td>
<td>12</td>
</tr>
<tr>
<td>2.4 3D Model Stability Analysis</td>
<td>13</td>
</tr>
<tr>
<td>3 RISC MOCAP ROS-PIXHAWK TESTING PLATFORM</td>
<td>14</td>
</tr>
<tr>
<td>3.1 Motion Capturing System</td>
<td>16</td>
</tr>
<tr>
<td>3.1.1 MOCAP Camera Position Optimization</td>
<td>16</td>
</tr>
<tr>
<td>3.1.2 MOCAP Camera Mounting Structure</td>
<td>18</td>
</tr>
<tr>
<td>3.1.3 MoCap MarkerSet and States Estimation</td>
<td>23</td>
</tr>
<tr>
<td>3.2 Quadrotor UAV with Onboard ROS-Pixhawk</td>
<td>26</td>
</tr>
<tr>
<td>3.2.1 Quadrotor Design Optimization</td>
<td>27</td>
</tr>
<tr>
<td>3.2.2 Quadrotor Construction</td>
<td>29</td>
</tr>
<tr>
<td>4 TRAJECTORY SHAPING IMPLEMENTATION</td>
<td>38</td>
</tr>
<tr>
<td>4.1 Simulation Implementation</td>
<td>38</td>
</tr>
<tr>
<td>4.1.1 MATLAB-Simulink Model</td>
<td>40</td>
</tr>
<tr>
<td>4.2 Hardware Implementation</td>
<td>40</td>
</tr>
<tr>
<td>4.2.1 Hardware Components</td>
<td>41</td>
</tr>
<tr>
<td>5 RESULTS</td>
<td>42</td>
</tr>
<tr>
<td>5.1 Simulation Results</td>
<td>42</td>
</tr>
<tr>
<td>5.1.1 Flat Circular Trajectory</td>
<td>42</td>
</tr>
<tr>
<td>5.1.2 Inclined Figure-8 Trajectory</td>
<td>42</td>
</tr>
<tr>
<td>5.2 Hardware Results</td>
<td>45</td>
</tr>
</tbody>
</table>
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Root Mean Square (RMS) of position error for simulation data</td>
<td>49</td>
</tr>
<tr>
<td>5.2</td>
<td>Root Mean Square (RMS) of position error for hardware test data</td>
<td>50</td>
</tr>
<tr>
<td>5.3</td>
<td>Root Mean Square (RMS) of position error for hardware test data after convergence of vehicle path to target path</td>
<td>52</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>5</td>
</tr>
<tr>
<td>2.2</td>
<td>7</td>
</tr>
<tr>
<td>2.3</td>
<td>10</td>
</tr>
<tr>
<td>3.1</td>
<td>17</td>
</tr>
<tr>
<td>3.2</td>
<td>18</td>
</tr>
<tr>
<td>3.3</td>
<td>19</td>
</tr>
<tr>
<td>3.4</td>
<td>19</td>
</tr>
<tr>
<td>3.5</td>
<td>20</td>
</tr>
<tr>
<td>3.6</td>
<td>21</td>
</tr>
<tr>
<td>3.7</td>
<td>21</td>
</tr>
<tr>
<td>3.8</td>
<td>22</td>
</tr>
<tr>
<td>3.9</td>
<td>23</td>
</tr>
<tr>
<td>3.10</td>
<td>24</td>
</tr>
<tr>
<td>3.11</td>
<td>25</td>
</tr>
<tr>
<td>3.12</td>
<td>28</td>
</tr>
<tr>
<td>3.13</td>
<td>28</td>
</tr>
<tr>
<td>3.14</td>
<td>29</td>
</tr>
<tr>
<td>3.15</td>
<td>31</td>
</tr>
<tr>
<td>3.16</td>
<td>31</td>
</tr>
</tbody>
</table>
3.17 X2208 1800Kv, 195W BLDC Motor ........................................ 32
3.18 Afro HV 20A ESC ................................................. 32
3.19 HQ 7X4.5 MM FIBERGLASS COMPOSITE PROP .................. 33
3.20 Hexagonal pipe of 16mmX18.5mmX1mm ......................... 33
3.21 FrSky 2.4GHz ACCST TARANIS X9D PLUS and X8R Combo Digital Telemetry Radio System (Mode 2) .................. 34
3.22 Lithium-Polymer Battery 2200mah, 11.1V, 35C ................... 34
3.23 Castle BEC PRO V2 ............................................ 35
3.24 FTDI Usb to TTL Serial Adapter Module ......................... 35
3.25 N150 Wi-Fi Nano USB Adapter ................................. 36
3.26 Motor tag numbers along with direction of rotation ............ 36
3.27 Quadrotor UAV View-1 ........................................... 37
3.28 Quadrotor UAV View-2 ........................................... 37
4.1 MATLAB-Simulink model used to implement 3D TS .............. 40
4.2 RISC MoCap RPTP hardware loop. .............................. 41
5.1 Circular trajectory plot from simulation results ................ 43
5.2 Control efforts commanded by controllers ...................... 43
5.3 Magnitude of velocities achieved by TS and DF vehicles ......... 44
5.4 Positional error generated by both TS and DF vehicles .......... 44
5.5 Figure-eight trajectory plot from simulation results .......... 45
5.6 Control efforts commanded by controllers ...................... 46
5.7 Magnitude of velocities achieved by TS and DF vehicles ......... 46
5.8 Positional error generated by both TS and DF vehicles .......... 47
5.9 Circular trajectory plot from hardware results ................ 48
5.10 Control efforts commanded by controllers ...................... 49
5.11 Magnitude of velocities achieved by vehicle using TS and DF controllers . . 49
5.12 Positional error generated by vehicle using TS and DF controllers . . . . . 50
5.13 Figure-eight trajectory plot from simulation results . . . . . . . . . . . . . . 50
5.14 Control efforts commanded by controllers . . . . . . . . . . . . . . . . . . . . . 51
5.15 Magnitude of velocities achieved by TS and DF vehicles . . . . . . . . . . . 51
5.16 Positional error generated by both TS and DF vehicles . . . . . . . . . . . . 52
### ACRONYMS

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>TS</td>
<td>Trajectory Shaping</td>
</tr>
<tr>
<td>DF</td>
<td>Differential Flatness</td>
</tr>
<tr>
<td>PP</td>
<td>Pure Pursuit</td>
</tr>
<tr>
<td>WP</td>
<td>Way Point</td>
</tr>
<tr>
<td>IR</td>
<td>Infra-Red</td>
</tr>
<tr>
<td>FOV</td>
<td>Field of View</td>
</tr>
<tr>
<td>UAV</td>
<td>Unmanned Aerial Vehicle</td>
</tr>
<tr>
<td>ESC</td>
<td>Electronic Speed Controller</td>
</tr>
<tr>
<td>BEC</td>
<td>Battery Eliminator Circuit</td>
</tr>
<tr>
<td>VTOL</td>
<td>Vertical Take-off and Landing</td>
</tr>
<tr>
<td>RISC</td>
<td>Robust Intelligent Sensing and Control</td>
</tr>
<tr>
<td>RPTP</td>
<td>ROS-PixHawk Testing Platform</td>
</tr>
<tr>
<td>ROS</td>
<td>Robot Operating System</td>
</tr>
<tr>
<td>PID</td>
<td>Proportional Integrator and Derivative</td>
</tr>
<tr>
<td>LQR</td>
<td>Linear Quadratic Regulator</td>
</tr>
<tr>
<td>IMU</td>
<td>Inertial Measurement Unit</td>
</tr>
<tr>
<td>SONAR</td>
<td>Sound Navigation and Ranging</td>
</tr>
<tr>
<td>LIDAR</td>
<td>Light Detection and Ranging</td>
</tr>
<tr>
<td>LOS</td>
<td>Line Of Sight</td>
</tr>
<tr>
<td>MoCap</td>
<td>Motion Capturing System</td>
</tr>
<tr>
<td>USU</td>
<td>Utah State University</td>
</tr>
</tbody>
</table>
CHAPTER 1
INTRODUCTION

Navigation of UAVs through the urban environment has acquired a significant attention of the researchers nowadays. Unlike other aerial vehicles, platforms like VTOL aircraft have become one of the major interest areas of researchers in the field of control and guidance. Use of quadrotors as a research platform is spread widely over the globe because of their simplicity in design and controls unlike traditional helicopters, which have complex mechanical linkages and variable blade pitch mechanisms. Unlike fixed wing UAVs, a multirotor can translate in any direction, which makes it the most desirable platform for navigation in small spaces [1]. This kind of UAV is easy to maintain because it can be repaired and deployed quickly after multiple crashes. Moreover, it is agiler as compared to fixed wing aircraft.

Since the urban environment includes various constraints like tall buildings, trees, static and moving obstacles, the UAVs require more robust and efficient path following algorithms to achieve higher levels of autonomy. Way-point following is a standard guidance method for following a path where a finite number of waypoints are selected on the desired path. Therefore, the desired trajectory could be represented as a continuous curve in a plane generated by those way-points. The path following algorithm guides a UAV in such a way that it approximately passes through all the waypoints continuously on the desired path. Although the desired path is already known, waypoints do not provide the complete curvature information to the vehicle’s path following controller. Therefore, waypoint following is only approximate technique and it may not work accurately for highly curved trajectories.

1.1 Related Work

For the development of path following guidance algorithms for unmanned vehicles, several concepts of trajectory following have been developed and tested by researchers based
upon specific requirements of the mission. The concept of differential flatness (DF) has been used by Ferrin et al. [2] from BYU to develop an algorithm for following aggressive trajectories. The LQR (Linear Quadratic Regulator) controller has been used in the DF concept to track the current waypoint on the desired trajectory which outputs high control effort to minimize the error between states of the vehicle and virtual target. Exerting high control effort leads to excessive use of the on-board limited power available on a UAV, which leads to less endurance over trivial positional errors. In the field of Aerospace Engineering, missile guidance concepts are highly developed where many optimal guidance laws are proven for their efficiency. Missile guidance laws that are being used successfully for intercepting targets are also being utilized for the guidance of unmanned vehicle to chase the target while maintaining a constant distance from the target. Such kind of guidance technique, called Pure Pursuit (PP) has been used by Medagoda et al. [3] where the desired trajectory is generated as the function of time and controller sends commanded acceleration to the vehicle to follow the current waypoint to converge to the desired path in tail-chase mode. Using Pure Pursuit guidance while moving on a curved path, the controller does not consider the target’s heading information and generates a heading error while moving towards the current waypoint. This heading error causes the vehicle to converge to an offset from the target trajectory, which makes it not an optimal law to trace the curved paths.

1.2 Research Contribution

To overcome the problems of vehicle convergence to positional offset errors and optimization of controller commands, trajectory shaping (TS) guidance [4] was chosen to be an optimal guidance technique. TS considers sufficient information about target’s states to generate optimal commanded lateral acceleration for the vehicle to accurately converge to target trajectory. The main contribution of this research is to develop an optimal 3D TS controller from existing 2D controller and conduct a comparative study between TS and DF for optimal convergence of vehicle path to desired target trajectory. The convergence of 3D TS guidance algorithm to desired target trajectory is proven by conducting mathematical analysis in Chapter-2 and confirmed with simulation and hardware results in Chapter-4.
To practically implement TS and DF guidance algorithms on hardware, the development of an accurate and reliable test platform is the secondary contribution of this research study to RISC lab. The development of this testing platform can help the other student lab members to test their initial work in research areas like visual inspection, collaborative localization in GPS-denied environments and much more. For achieving this goal, a test platform was designed and established in the RISC lab at USU, by using a MoCap system and Robot Operating System (ROS)-PixHawk integration on a small quadrotor having an on-board computer. The contribution to design and specifications of RISC ROS-PixHawk Testing Platform (RISC RPTP) will be described briefly in Chapter-3 of this thesis. An international conference paper was submitted to ICUAS-2016, based on the comparative study of optimal path convergence using 3D trajectory shaping (TS), DF and pure pursuit (PP) simulation data. Simulation results were taken by implementing these guidance algorithms in MATLAB-Simulink. A quadrotor UAV model described in Chapter-2 was mapped to accept the output commanded accelerations generated by these algorithms to get the simulation results data.
CHAPTER 2

TRAJECTORY SHAPING GUIDANCE

In this chapter, the original 2D model of TS is described along with the explanation of its convergence to desired target trajectory under certain necessary conditions. The 3D model is derived by adopting the basic 2D controller model into a horizontal and vertical planes of 3D Cartesian coordinate frame. The lateral accelerations commanded in horizontal and vertical planes are then transformed into the form of components of acceleration required to be commanded to Quadrotor model along the three axes of 3D co-ordinate system. The longitudinal motion of the vehicle is controlled by using a simple proportional velocity controller. A proportional gain is directly multiplied with the difference of vehicle’s commanded velocity and instantaneous velocity of vehicle achieved while moving towards converging to the target trajectory. Angular heading rates method used for this transformation of accelerations is described briefly in this chapter. A quadrotor model employed by Ferrin et al. [2] to implement DF controller, is adopted in this research to implement 3D TS guidance model. This quadrotor model accepts the inputs in the form of outputs commanded by 3D TS guidance model. In the last section of this chapter, the mathematical analysis proves the convergence of 3D TS guidance model to the desired target trajectory.

2.1 Trajectory Shaping 2D Model

The concept of Trajectory Shaping (TS) guidance was developed by Ratnoo et al. [4] from missile guidance laws to follow the desired path by maintaining an approximately constant distance between the virtual target and a UAV. In this guidance method, the acceleration is applied to the vehicle in the direction lateral to its velocity vector and pointing towards LOS vector. LOS angle is the angle of a vector from the vehicle position to the target position, measured from the horizontal axis. The commanded lateral acceleration is generated as the function of the vehicle heading angle, target-heading angle and line-of-
sight (LOS) angle. Trajectory Shaping guidance has been proposed as an optimal guidance technique for tracing a curved path as it utilizes minimum control effort. In this concept, the vehicle path has the same instantaneous curvature as that of the virtual target path. Instead of forcing itself to be on the target waypoint, UAV rather utilizes its control effort to be on the target trajectory at any instant of time. Trajectory Shaping guidance has a wide vehicle launch envelope which allows the target path following to work efficiently for a UAV to be launched even from deviated initial launching conditions. At arrival to the target’s path, the vehicle path is shaped in such a way that it enters smoothly into its trajectory without overshoot. The convergence rate of vehicle’s trajectory to the desired path in this approach is twice than that of Pure Pursuit guidance [4]. The general 2D geometry of TS model is shown in Fig. 2.1.

![Fig. 2.1: 2D Geometry for Trajectory Shaping](image)
The general 2D guidance model is given by,

\[ a_c = \frac{V_d^2}{R_{xy}} (6\lambda - 4\alpha_v - 2\alpha_t), \]

\[ \therefore a_c = \frac{V_d^2}{R_{xy}} (4(\lambda - \alpha_v) + 2(\lambda - \alpha_t)), \]

where,

\[ R_{xy} = \sqrt{(x_t - x_v)^2 + (y_t - y_v)^2}, \]

and, \( V_d \) is the desired velocity of the vehicle, \( \lambda \) is the LOS angle while \( \alpha_v \) and \( \alpha_t \) is the heading angle for the vehicle and target respectively. The \( a_c \) is the lateral acceleration acting perpendicular to the vehicle velocity vector in the plane containing the LOS vector, velocity vectors of vehicle and target as shown in Fig.2.1.

### 2.2 3D TS Model and Mapping To Quadrotor Commands

For the 3D model of TS guidance, all required 3D relative geometry between virtual target and the vehicle is shown in Fig. 2.2. For any predefined trajectory, the relative geometry vectors of target and vehicle motion, always remain on a single 2D plane. Therefore to develop a 3D model this 2D geometry set of vectors is adopted, and every single vector is resolved into its projections onto vertical and horizontal planes of a 3D coordinate frame. In this 3D geometry, all required angles are categorized in the form of azimuth angle and elevation angle defined in horizontal and vertical planes respectively. The required formulation is given by,

\[ V_{t_x} = V_t \cos \theta_t \cos \psi_t, \]

\[ V_{t_y} = V_t \cos \theta_t \sin \psi_t, \]

\[ V_{t_z} = V_t \sin \theta_t, \]
Fig. 2.2: Relative geometry between virtual target moving on a pre-defined trajectory and the UAV

\[ V_{v_x} = V_v \cos \theta_v \cos \psi_v, \quad (2.1) \]

\[ V_{v_y} = V_v \cos \theta_v \sin \psi_v, \quad (2.2) \]

\[ V_{v_z} = V_v \sin \theta_v, \quad (2.3) \]

Here, \( \hat{V}_t, \hat{V}_v \) are target and vehicle velocity vectors with their components in \( x, y \) and \( z \) directions.
The azimuth angles $\psi_l$, $\psi_v$ and $\psi_t$ for LOS vector, vehicle heading, and target heading respectively are given by,

\[
\psi_l = \tan^{-1}\left(\frac{y_t - y_v}{x_t - x_v}\right),
\]
\[
\psi_v = \tan^{-1}\left(\frac{V_{V_v}}{V_{V_{x_v}}}\right),
\]
\[
\psi_t = \tan^{-1}\left(\frac{V_{V_t}}{V_{V_{x_t}}}\right).
\]

The elevation angles $\theta_l$, $\theta_v$ and $\theta_t$ for LOS vector, vehicle heading and target heading respectively are given by,

\[
\theta_l = \frac{(z_t - z_v)}{R_{xy}},
\]
\[
\theta_v = \tan^{-1}\left(\frac{V_{V_v}}{V_{V_{z_v}}/R_{xy}}\right),
\]
\[
\theta_t = \tan^{-1}\left(\frac{V_{V_t}}{V_{V_{z_t}}/R_{xy}}\right),
\]

where, $V_{v_{x,y}}$, $V_{t_{x,y}}$ and $R_{xy}$ are the 2D components of $V_v$, $V_t$ and $R$ respectively in XY plane and are given by,

\[
V_{v_{x,y}} = \sqrt{V_{V_{x}}^2 + V_{V_{y}}^2},
\]
\[
V_{t_{x,y}} = \sqrt{V_{V_{t}}^2 + V_{V_{y}}^2},
\]
\[
R_{xy} = \sqrt{(x_t - x_v)^2 + (y_t - y_v)^2}.
\]

The commanded lateral accelerations $a_v^h$ and $a_v^v$ to vehicle in horizontal and vertical planes respectively can be computed as,

\[
a_v^h = \frac{V_d^2}{R_{xy}} \left(4 (\psi_l - \psi_v) + 2 (\psi_l - \psi_t)\right),
\]
\[
a_v^v = \frac{V_d^2}{R} \left(4 (\theta_l - \theta_v) + 2 (\theta_l - \theta_t)\right),
\]
where, $R$ is given by,

$$R = \sqrt{(x_t - x_v)^2 + (y_t - y_v)^2 + (z_t - z_v)^2}.$$  \hfill (2.4)

The vehicle heading rates and velocity controller are given by,

$$\dot{\psi}_v = \frac{a^{h}_v}{V_v^2 \cos \theta_v},$$

$$\dot{\theta}_v = \frac{a^{v}_v}{V_v},$$

$$\dot{V}_v = k_v (V_d - V_v),$$

$$\dot{V}_{v,x,y} = k_v (V_d - V_{x,y}),$$

where, $V_d$ and $k_v$ are vehicle’s desired velocity and velocity controller proportional gain respectively. To compute the components of the quadrotor commanded acceleration along the $X$, $Y$ and $Z$ axis of 3D coordinate frame, vehicle velocity vector components given by Eq. 2.1, 2.2 and 2.3 which are differentiated with respect to time and resultant acceleration components are given by,

$$\ddot{x}_v = V_{v,x,y} c_{\theta_v} c_{\psi_v} - \dot{\theta}_v V_{v,x,y} s_{\theta_v} c_{\psi_v} - \dot{\psi}_v V_{v,x,y} c_{\theta_v} s_{\psi_v},$$

$$\ddot{y}_v = V_{v,x,y} c_{\theta_v} s_{\psi_v} - \dot{\theta}_v V_{v,x,y} s_{\theta_v} s_{\psi_v} + \dot{\psi}_v V_{v,x,y} c_{\theta_v} c_{\psi_v},$$

$$\ddot{z}_v = \dot{\theta}_v V_v c_{\theta_v} + \dot{V}_v s_{\theta_v} + g,$$

where $c_{\theta_v}$, $s_{\theta_v}$ and $g$ represent the $\cos(\theta_v)$, $\sin(\theta_v)$ and acceleration due to gravity respectively. The vector of commanded accelerations $u_p$, which are sent to quadrotor model to make it follow the target trajectory is given by,

$$u_p = \begin{bmatrix} 
\ddot{x}_v \\
\ddot{y}_v \\
\ddot{z}_v \end{bmatrix}.$$
2.3 Quadrotor Model

The UAV’s model was derived using the ‘X’ Quadrotor configuration, shown in Fig. 2.3 and the equations of motion given in [2]. The quadrotor model has 12 states. Therefore the mathematical formulation is derived in state-space form to keep the input vector easy to map into guidance algorithm’s commanded acceleration components. The 12 state vector of Quadrotor UAV is given by,

\[
X = [x_v, y_v, z_v, \dot{x}_v, \dot{y}_v, \dot{z}_v, \phi, \theta, \psi, p, q, r]^T,
\]

(2.5)

here, \(x_v, y_v\) and \(z_v\) are vehicle positions along \(x, y\) and \(z\)-axis of 3D inertial coordinate frame. The next three states \(\dot{x}_v, \dot{y}_v\) and \(\dot{z}_v\) are the velocity components along three axes of inertial frame. Next three are Euler angles \(\phi, \theta\) and \(\psi\) which are denoted by roll, pitch and yaw respectively in the body frame of the vehicle. The last three states \(p, q,\) and \(r\) are the angular rates in body frame.

The UAV is assumed to have an onboard autopilot (PID attitude controller) to eliminate the need for controlling the moments by original guidance controller. Therefore, the...
The state vector in Eq. 2.5 reduces to,

\[ X = [x_v, y_v, z_v, \dot{x}_v, \dot{y}_v, \dot{z}_v, \phi, \theta, \psi]^T. \]

The onboard controller is designed to accept the commands in the form of an input vector given by,

\[ v = [T, \phi, \theta, r]^T, \]

where, \( T \) is the commanded thrust, \( \phi \) and \( \theta \) are commanded roll and pitch angles respectively, while \( r \) is the commanded yaw rate in body frame. These commands are generated by using an inverse mapping technique explained in subsection 2.3.1.

The state-space equations for acceleration components are given by,

\[
\begin{bmatrix}
\dot{x}_v \\
\dot{y}_v \\
\dot{z}_v
\end{bmatrix} = -\frac{T}{m} \begin{bmatrix}
c_\phi s_\theta c_\psi + s_\phi s_\psi \\
c_\phi s_\theta s_\psi - s_\phi c_\psi \\
c_\phi c_\theta
\end{bmatrix} + \begin{bmatrix}
0 \\
0 \\
g
\end{bmatrix},
\]

\[ \dot{\psi} = p \frac{s_\phi}{c_\theta} + r \frac{c_\phi}{c_\theta}, \]

\[
\begin{bmatrix}
\dot{x}_v \\
\dot{y}_v \\
\dot{z}_v
\end{bmatrix} = u_p + \begin{bmatrix}
0 \\
0 \\
g
\end{bmatrix},
\]

\[ \dot{\psi} = u_\psi, \]

where, the input acceleration vector \( u_p \) and yaw rate component \( u_\psi \) are given by,

\[ u_p = -\frac{T}{m} \begin{bmatrix}
c_\phi s_\theta c_\psi + s_\phi s_\psi \\
c_\phi s_\theta s_\psi - s_\phi c_\psi \\
c_\phi c_\theta
\end{bmatrix}, \]

\[ u_\psi = q \frac{s_\phi}{c_\theta} + r \frac{c_\phi}{c_\theta}. \]
The combined input acceleration vector is,

\[ u = \begin{bmatrix} u_p \\ u_\psi \end{bmatrix} = \begin{bmatrix} c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\phi s_\theta s_\psi - s_\phi c_\psi \\ c_\phi c_\theta \\ p \frac{s_\phi}{c_\theta} + r \frac{c_\phi}{c_\theta} \end{bmatrix} \]

### 2.3.1 Inverse Mapping

Inverse mapping is given in [2], is a technique used for mapping the UAV’s input vector of throttle \((T)\), roll \((\phi)\), pitch \((\theta)\) and yaw rate \((r)\) to the acceleration vector commanded by TS guidance controller. The basic formulation is given by,

\[ v_c = [T_c, \phi_c, \theta_c, r_c]^T, \]

For computing thrust input,

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

where, \(R(\phi), R(\theta)\) and \(R(\psi)\) are the Quadrotor’s rotation matrices.

By normalizing the three components of \(u_p\), the commanded thrust is given by,

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

For \(\phi_d\) and \(\theta_d\), a new vector \(A\) is defined as,

\[
A = \begin{bmatrix} A_1 \\ A_2 \\ A_3 \end{bmatrix} = R(\psi)u_p \begin{bmatrix} m \\ -T \end{bmatrix} = R^T(\theta)R^T(\phi) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.
\]
Solving for $\phi$ and $\theta$, commanded roll and pitch are given by,

$$\phi_c = \sin^{-1}(-A_2),$$
$$\theta_c = \tan^{-1}\left(\frac{A_1}{A_3}\right).$$

From $u_\psi$, commanded yaw rate can be computed as,

$$r_c = u_\psi \cos(\theta) \cos(\phi) - \dot{\theta} \sin(\phi),$$

where, $\dot{\theta} = q$, which can be measured directly from an onboard IMU or can be computed by differentiating $\theta_d$ with respect to sampling time.

### 2.4 3D Model Stability Analysis

As shown in Fig.2.2 and previously explained in section-2.2 of this Chapter, the 3D model was developed by adopting and resolving the relative geometry of TS model from a 2D to 3D coordinate frame. Therefore if it can mathematically be proved that TS model will converge to the desired path and be stable in the 2D coordinate frame, it will be stable in the 3D coordinate frame as well. The convergence of 2D TS guidance model to the desired path and being stable there, has successfully been proved by Ishmaal T. Erekson [5].

For this analysis it is assumed that virtual target is moving on a constant path of some radius $r$ and a vehicle is following it with constant velocity $V_c$. The vehicle will converge to the desired path and maintain a constant LOS distance $R^*$ from virtual target for any $V_c \in (0, \infty)$, $r \in [(-\infty, 0) \cup (0, \infty)]$ and $R^* \in (0, 2r)$. 
CHAPTER 3
RISC MOCAP ROS-PIXHAWK TESTING PLATFORM

For the successful execution of an autonomous mission, the autonomous vehicle has to have a robust controller design and also some reliable feedback system which can provide the sensor data at considerably good rates with high precision and accuracy. The reliable sensor feedback required is as important as the controller design is considered. The type of sensors and feedback data required to be provided to the controller is always specific to the mission type. For example, for a small UAV or an autonomous ground vehicle, the desired task could be to follow a specific path or could be avoiding the collision with an obstacle while moving on a desired path. For both of these kinds of autonomous missions, the feedback information required from sensors could be the position and velocity vectors of vehicle and obstacle in an inertial frame. The sensors used to get the feedback data for these missions potentially could be GPS [6], an onboard IMU, pitot tube, SONAR, LIDAR or an optical camera for detecting a visible obstacle.

For this research, the guidance algorithm adopted has originally been developed from missile guidance laws. The type of sensor data required for this kind of guidance law is the 3D position and velocity components of the vehicle in the inertial frame along with the angular position and angular rates of the vehicle about its body axis while it is in motion. A quadrotor UAV was selected for the practical implementation of this guidance algorithm. This UAV is selected with an onboard autopilot to control the attitude for maintaining a stable and level horizon during flight. The selected autopilot has its IMU sensor which provides feedback to its Proportional, Integral, Derivative (PID) controller to stabilize vehicle’s horizon level. This autopilot accepts the commands from the guidance controller in the form of a vector, which consists of commanded thrust, roll angle, pitch angle, and yaw rate to achieve the commanded position and velocity in the inertial frame. This autopilot solves the problem of stabilizing the UAV and also accepting the commands
from the guidance controller in the desired form. For the position and velocity information of a moving vehicle in an inertial frame, various type of sensors and techniques can be used. A technique known as dead reckoning could be used to estimate the position and velocity of a UAV by using Kalman filters on raw IMU data. However, this method will require very accurate IMU sensors (Gyros and Accelerometers) with known biases and errors, which makes these sensors to be cost-wise very expensive. Even if the higher cost of these sensors is traded for the certainty of their accuracy and biases, it would be used on a single vehicle at a time which makes it even more expensive to have it for each one of those multiple vehicles on a mission. Another approach of getting the vehicles position and velocity data is to use outdoor GPS sensor on the UAV.

The primary limitations of using this sensor are that it gives the required data not more than 1Hz along with the positional accuracy only up to several meters of distance. Also, GPS sensors cannot be used for indoor testing of guidance algorithms, as it would become potentially dangerous for testing of the controller during initial development phase in the presence of humans, public and national property. All of these vital factors described above points strongly towards having a system that can provide the required feedback data for a UAV at considerably higher data rates and accuracy even while testing indoors within constrained environment without harming humans or any public property. Only then the cost of having such an expensive system would be justified if it can be used for multiple UAVs and also again and again without getting affected by vehicles multiple crashes during the initial testing of guidance controllers development phase. After a long study about existing methods of getting sensor data, motion capturing system (MoCap) was used to get very accurate data at very high rates. MoCap along with ROS-PixHawk integration was used to sense the states of UAV, computing commands from guidance controller and then send these commands to UAV, complete the loop of sensing and controlling to follow the desired trajectory. The Motion Analysis IR Cameras and silver coated marker system was used to design and establish a MoCap system in RISC Lab at USU. In this chapter, the design and construction of this MoCap system are explained thoroughly along with the
design and construction of a quadrotor UAV used for this research.

3.1 Motion Capturing System

Motion Capturing is a technique of sensing the motion of the desired object by using IR cameras. IR LEDs mounted on the IR sensing cameras then project IR light onto reflective marker set mounted on the object whose motion is required to be captured. IR sensing cameras then sense the motion whenever that object moves within the camera field of view. The markers used to reflect the IR light are mostly spherical in shape so that no matter what orientation they have with respect to the camera, the light will get reflected back to it. Minimum two cameras are required to sense and estimate the 3D motion of any object by triangulating the geometry of camera positions. The necessary coverage volume can be covered by just two to any number of cameras depending upon camera specifications and their position geometry. These cameras then transfer their captured data through cables to an Ethernet switch, where all the information coming from multiple cameras is mixed and then, sent to the main computer which runs the motion computation software.

The general structure of MoCap system is shown in Fig.3.1. This system can capture and process the motion at very high rates usually at around 200 times in a second. This rate varies according to what different manufacturers can provide according to the various models available. For establishing this MoCap system at USU, the camera system was obtained from a company named as 'Motion Analysis.' The system was designed to have 16 IR cameras to cover the total volume of 4.8m x 3.9m x 2.5m (L x W x H) of volume. From Motion Analysis, Osprey IR Camera was used which has 0.3 megapixels (640x480 pixels) and can capture the motion at the maximum of 245 frames per second. Further information about this camera can be found at [7]. The motion computation software called 'Cortex' was used from Motion Analysis, which can be found at [8].

3.1.1 MOCAP Camera Position Optimization

The required capture volume was calculated based on the space available inside RISC lab room at USU, where the MoCap system was planned to be installed. This room had
sufficient space to accommodate the cubical volume spanning 4.8mx3.9mx2.5m (LxWxH), for mounting 16 cameras. The camera location pitch was selected in such a way that cameras are mounted equidistant from each other, on the four edges of the mounting structure. The camera position geometry for 16 IR cameras is shown in Fig. 3.2. The 3D isometric view of the camera layout along with workspace is shown in Fig. 3.3. The cameras were mounted on the frame with an orientation in such a way, that all the edges of the field of view remain inside the required capture volume. This method was used to ensure that camera field of view can be utilized optimally. The top view of layout in Fig. 3.3 and Fig. 3.5 shows that how the camera on corner and edge of the mounting structure should be pointing their fields of view to for optimal use. The isometric view of the camera’s field of view is shown in Fig. 3.6 along with three axes of Cartesian coordinate frame system. All the 16 cameras were mounted on the structure using this technique and Fig. 3.7 shows the intersection of all camera FOVs. Fig. 3.8 defines the volume covered by all the camera FOVs together. The intersection density of FOV of all the cameras is highest in the middle of the covered volume. Intersection density of FOV means the number of FOVS from different cameras

Fig. 3.1: General MoCap system structure
intersecting or merging at a single point. In this configuration, every single point is visible to at least four cameras.

![Top view of IR camera layout](image)

**Fig. 3.2: Top view of IR camera layout**

### 3.1.2 MOCAP Camera Mounting Structure

Holding the cameras in fixed position with considerable mutual rigidity and also having the ability to dampen vibrations while capturing the motion of a vehicle, are critical design factors. After setting the cameras on any mounting structures, the whole system has to be calibrated so that the cortex software can calculate triangulations for all the cameras and use this triangulated geometry to estimate the states of any detected marker within the captured volume. Having any relative motion between cameras or vibrations in the structure leads to massive errors in the marker states estimation. Due to this relative movement and vibrations, the mutual triangulation geometry gets distorted which gives rise to errors and wrong data. For fixing these issues, the structure has to be considerably rigid to counter
any motion or vibrations. After considering all these vital factors, instead of mounting the cameras on individual mounts or tripods, a single structure design was selected for this
Fig. 3.5: Top view of layout showing the FOV orientation of camera mounted on an edge project. Teak wood was chosen for the construction of the mounting structure, as it is lighter in weight and have better vibration absorption capacity than metals. It possessed enough strength to construct a structure that can hold the cameras with required rigidity. Rectangular section beams and columns of 4inchx2inch were used for this design. Two rectangular frames of 4.8mX3.9m (LxW) for bottom and top were constructed. Eight columns of the same material were mounted on the bottom frame to support the weight of top side frame. Wood glue, screws, and sheet metal strips were used to fasten all parts together. The finished structure is shown in Fig.3.9.

The flooring for the captured volume was provided by using a black colored rubber foam of about half an inch of thickness. This carpet served the purpose of preventing the floor to reflect IR light back to cameras. Otherwise, it would have added noise to the system. The secondary benefit of this flooring is that it can save an aerial vehicle from having any minor crash impacts from the hard floor. During the implementation and initial testing of any new controller on a vehicle for the autonomous mission, a couple of crashes can always be expected. The foam flooring could be beneficial to save a UAV from minor
Fig. 3.6: Isometric view of 3D XYZ co-ordinate frame along with the FOV orientation of camera mounted on a corner

Fig. 3.7: Isometric view of layout showing the intersections of all the cameras FOV.
impacts but for harder crashes, another safety measure was strongly required to protect expensive hardware on the UAV from damages. This safety measure would reduce the UAV’s downtime as well as the effort for repairing it to fly again. For this purpose, a bottom net mechanism was designed where after the UAV takes off, the net can be raised about a maximum of 2 feet from the ground by ropes and pulleys mechanism. A wooden frame made of 2inchX1inch rectangular section wood was used to help stretching the bottom net for having the strength to save the UAV from crashing to the floor. Eight ropes from the main ratchet pulley mounted on the main structure, were guided by through smaller pulleys to the hinge points on the bottom net wooden frame. The guided rope mechanism is shown in Fig.3.10. The primary pulley has a hand lever which is used either to raise or to lower the bottom net manually. The net is held in raised position by locking the non-return ratchet on the primary pulley. This whole MoCap camera mounting structure encloses another net that is hanging from the roof to provide the safety to people working in the lab and to the IR cameras from UAV’s crashes while conducting experiments. For entering into the workspace, this side netting was provided with two openings on opposite
sides, which can be closed after placing the UAV inside and then start the experiment.

Fig. 3.9: MoCap Camera mounting structure

3.1.3 MoCap MarkerSet and States Estimation

For getting the feedback data from MoCap system, couples of IR light reflective markers are used to design an asymmetric marker set structure, which is mounted on top of Quadrotor UAV. These markers can directly be installed on the UAV if the structure of the vehicle is rigid enough to maintain the constant mutual geometry of the markers. Once the marker set has is mounted on the UAV, its recognizing template is created in the cortex software, which will save the mutual geometry of all the markers. For making a recognizing template for its corresponding marker set, all the required instructions can be found at [8]. A particular name is assigned to this template, and whenever any experiment is run, this model is loaded in the workspace of cortex software. When experimented is run, cortex starts recognizing the vehicle from the particular marker set mounted on it. A simple
Fig. 3.10: Rope Guide Mechanism
marker set mounted on X-configuration Quadrotor UAV has been shown in Fig. 3.11. In this marker set, four markers A, B, C, and D have been mounted asymmetrically on the UAV so that its heading is always known. The total of 12 states of a UAV can be obtained from this MoCap system. Eq. 2.5 in chapter-2 gives the states vector of a vehicle. Cortex provides only the 3D position of every single marker. Therefore the rest of other 6 states have to be computed by using the mathematical formulation for vector geometry of the marker set. For simplicity, the marker D can directly be used to get the 3D position and velocity of UAV in XYZ coordinate frame.

![Fig. 3.11: A Marker Set Geometry Mounted on X-Quadrotor in XY-Plane](image-url)
The last six states of the vehicle, for example, can be computed from vector geometry in vehicle’s body frame as,

\[
\phi = \tan^{-1}\left(\frac{B_z - A_z}{B_y - A_y}\right),
\]

\[
\theta = \tan^{-1}\left(\frac{B_z - C_z}{B_x - C_x}\right),
\]

\[
\psi = \tan^{-1}\left(\frac{B_y - C_y}{B_x - C_x}\right),
\]

and,

\[
p_{n+1} = \frac{\phi_{n+1} - \phi_n}{T_s},
\]

\[
q_{n+1} = \frac{\theta_{n+1} - \theta_n}{T_s},
\]

\[
r_{n+1} = \frac{\psi_{n+1} - \psi_n}{T_s},
\]

where, \(A_x\), \(A_y\) and \(A_z\) are X, Y and Z positions of marker A in vehicle’s XYZ body frame. Same notations are used for all the markers used for this set. \(T_s\) is the sampling time and \(p_{n+1}\), \(q_{n+1}\) and \(r_{n+1}\) defines the equations to compute \(p\), \(q\) and \(r\) in discrete time method.

### 3.2 Quadrotor UAV with Onboard ROS-Pixhawk

A quadrotor [9] uses two clockwise (CW) and two counter-clockwise (CCW) pairs of fixed pitch propellers. Motor speeds of each rotor are varied independently to achieve the total control and maneuverability in all possible directions. Quadrotor design is preferred over conventional the helicopter rotor design in which each rotor blade can change it’s pitch dynamically while moving around the main rotor. At the smaller scale, a quadcopter is an optimal choice for research purposes as the design is more durable than that of conventional helicopters due to its mechanical simplicity. Quadrotors use relative smaller blades which possess less kinetic energy thereby reducing the risk to cause bigger damage. At the same
time, smaller blades need higher spinning speed to produce same amount thrust. Therefore, the design simplicity comes at the cost of efficiency. Also, smaller rotors, spinning at very high speeds generates very high-frequency vibrations inside the quadrotor’s body, which can cause damage to autopilot’s sensors. But this problem can be solved by using relatively bigger blade size rotors and increasing blade size would improve the efficiency, as it would take lesser power to generate the same thrust. The onboard power can be used more efficiently by spinning a bigger blade to move a larger volume of air at a slower speed than by using a smaller blade to move a smaller volume of air at relatively higher spinning speed. Therefore, motors and propeller combination should be selected in such a way that the efficiency of conversion of electric power into thrust is reached to its maximum.

For the testing and implementation of control algorithms, a robust and reliable platform is required. The primary design parameters important to be considered for a UAV are hover flight time, thrust to weight ratio. Secondly, the mechanical structure has to be well designed that it can tolerate many crashes without having a significant damage to its expensive components especially motors and sensors. Having multiple crashes during the initial development and testing of any controller are highly expected.

### 3.2.1 Quadrotor Design Optimization

The UAV designing process was started first by roughly measuring the total payload that was going to be carried by the quadrotor during flight. The whole RISC platform was chosen to be designed in such a way that all the 12 states of UAV will be measured by using MoCAp system. An onboard computer receives the 12 feedback states to generate the control commands. The control commands are sent to quadrotor’s autopilot through a wired connection. Therefore, the IR marker set, onboard computer, autopilot and all their electric accessories have to be considered as payload for the quadrotor.

For the selection of motors, propellers, ESCs and battery to be used for this design, an online available calculator known as ‘eCalc’ [10] was used for calculating the size and specifications of all these components. Based on the selected components for this design, the expected flight characteristics obtained are shown in Fig.3.12, Fig.3.13 and Fig.3.14
Fig. 3.12: Battery Specifications and Power Consumption

|------------------|-----------------|-----------------|------------------------|-----------------|---------------------------|-------------------------|---------------------------|--------------------------|----------------------|-----------|

Motor @ Optimum Efficiency
- Current: 8.73 A
- Voltage: 10.35 V
- Revolutions*: 17168 rpm
- Electric Power: 90.3 W
- Mech. Power: 76.7 W
- Efficiency: 84.9%

Fig. 3.13: Motor Power Consumption and Output

|------------------|------------------|-----------------|------------------------|-----------------------|-----------------------|--------------------------|---------|-----------------|----------------------|

|------------------|-----------------|-----------------|------------------------|---------------------|-----------------------|--------------------------|---------|-----------------|----------------------|

- °C: Celsius
- °F: Fahrenheit
3.2.2 Quadrotor Construction

The list of UAV’s basic components selected by using eCalc is given by,

- From '3dr.com', [11] PixHawk Autopilot was selected for controlling the attitude of Quadrotor and accepting commands from an onboard computer. The PixHawk autopilot is shown in Fig. 3.15.

- From ‘ameridroid.com’, [12] An onboard computer ODROID-XU4 [0007A] was selected for running guidance controller in ROS and sending commands to autopilot. The computer is shown in Fig. 3.16.

- From 'readytoflyquads.com', [13] X2208 1800Kv, 195 watts brushless dc motors were selected. The motor is shown in Fig. 3.17.

- From 'hobbyking.com', [14] Afro HV 20A ESCs were selected. The ESC is shown in Fig. 3.18.
From 'readytoflyquads.com', [15] HQ 7X4.5 MM FIBERGLASS COMPOSITE PROP were selected. These propellers are available in multiple colors. The propellers are shown in Fig.3.19.

From 'rockwestcomposites.com', [16] Hexagonal pipe of 16mmX18.5mmX1mm cross-section was used for making the arms of quadrotor UAV. The center plates were cut from carbon fiber fabric sheet of 1.5mm thickness. The hexagonal pipe is shown in Fig.3.20.

From 'hobbyking.com', [14] FrSky 2.4GHz transmitter and receiver set was used. The radio set is shown in Fig.3.21.

From 'Amazon.com', [17] Lithium-Polymer Battery 2200mah, 11.1V, 35C was selected. The battery is shown in Fig.3.22.

From 'castlecreations.com', [18] Castle BEC PRO V2 was selected for providing the power to onboard computer. BEC is shown in the Fig.3.23.

From 'Amazon.com', [18] FTDI Usb to TTL Serial Adapter Module was used to connect onboard Odroid computer to PixHawk. The FTDI Adapter is shown in Fig.3.24.

From 'Edimax.com', [19] N150 Wi-Fi Nano USB Adapter was used for communication of Odroid with the ground station running ROS-Master. The Wi-Fi adapter is shown in Fig.3.25.

The Quadrotor was constructed using the machine tools at RISC lab. Three center plates of 10mmX10mmX1.5mm carbon fiber material were used along with the hexagonal tubes to complete the frame structure. All four rotor arms were sandwiched in between two lower plates and fastened by using M2 nuts and bolts. Motors were mounted on arms using M3 bolts and lock-tight glue. The inner space in between these two plates was occupied by four ESCs and power distribution silicon wires. Motors are connected to ESCs by keeping the direction of rotation according to the mapping shown in Fig.3.26. Motor and ESCs
sets are tagged with numbers from 1-4 to recognize their connections with PixHawk’s signal output. ESC’s are connected to PixHawk’s first four main output ports according to motor’s tag number. PixHawk and FrSky radio receiver were mounted on top of the middle plate
by using memory foam pads to absorb the vibrations. The PixHawk’s both body axes were perfectly aligned to the both axes of UAV’s body frame to avoid IMU misalignment errors. Radio receiver’s S-bus port is connected to PixHawk RC connection port. Power is provided
to PixHawk by using its power module from 3dr.com. The top plate was mounted on the middle plate using four nylon plastic stand-offs to provide enough space for PixHawk wiring. The Odroid computer was mounted on this top plate using nylon nuts and bolts. Odroid
was provided with power from BEC and was connected to PixHawk using FTDI module for communication. A WiFi internet USB module is plugged into Odroid for communication with ground station computer which runs ROS-Master. The final version of UAV platform after completion of construction is shown in Fig. 3.27 and Fig. 3.28.
Fig. 3.23: Castle BEC PRO V2

Fig. 3.24: FTDI Usb to TTL Serial Adapter Module
Fig. 3.25: N150 Wi-Fi Nano USB Adapter

Fig. 3.26: Motor tag numbers along with direction of rotation
Fig. 3.27: Quadrotor UAV View-1

Fig. 3.28: Quadrotor UAV View-2
CHAPTER 4
TRAJECTORY SHAPING IMPLEMENTATION

4.1 Simulation Implementation

For the implementation of 3D TS guidance model in MATLAB-Simulink, a set of
equations of motion for virtual target were used to generate a circular trajectory in the 2D
XY plane. The set of equations is given by,

\[ V_t = V_c \frac{R^*}{R}, \]

\[ \omega = \frac{V_t}{r}, \]

\[ \omega_1 = \frac{\omega}{S_t}, \]

\[ \omega_2 = \omega, \]

\[ x_t = r \cos \omega_1 t, \]

\[ y_t = r \sin \omega_2 t, \]

\[ z_t = n, \]

\[ \dot{x}_t = -r \omega_1 \sin \omega_1 t, \]

\[ \dot{y}_t = r \omega_2 \cos \omega_2 t, \]

\[ \dot{z}_t = 0, \]

\[ \ddot{x}_t = -r \omega_1 \omega_1 \cos \omega_1 t, \]

\[ \ddot{y}_t = -r \omega_2 \omega_2 \sin \omega_2 t, \]

\[ \ddot{z}_t = 0, \]
where, $V_c$ is the desired velocity of vehicle. $R$ is the LOS distance between the virtual target and the vehicle positions provided by Eq.2.4. $R^*$ is the user defined constant LOS distance between the virtual target and vehicle. The main goal of minimizing the control effort generated by TS guidance controller is achieved by using a velocity controller to control the virtual target motion. Eq.4.1 gives the velocity controller. Whenever the LOS distance between target and UAV becomes less than $R^*$, virtual target’s speed increases relative to the UAV’s speed and $R$ becomes equal to $R^*$ to reach the equilibrium state. $\omega$, is the angular velocity required for the motion of target and $n$ is the desired altitude of trajectory. $S_t$ is the parameter which is used either to select a circular trajectory or a figure-eight trajectory for the virtual target. For selecting circular trajectory, $S_t = 1$ is used, while for figure eight, $S_t = 2$ is used.

The 3D trajectory is generated by rotating the 2D set of trajectory equations about either x or y-axis in XYZ coordinate frame. The 3D set of equations for trajectory along with rotation about x-axis is given by,

$$R_x = \begin{bmatrix}
1 & 0 & 0 \\
0 & c(\phi_x) & s(\phi_x) \\
0 & -s(\phi_x) & c(\phi_x)
\end{bmatrix},$$

$$\begin{bmatrix}
 x_{t3d} \\
y_{t3d} \\
z_{t3d}
\end{bmatrix} = R_x \begin{bmatrix}
x_t \\
y_t \\
z_t
\end{bmatrix},$$

$$\begin{bmatrix}
 \dot{x}_{t3d} \\
\dot{y}_{t3d} \\
\dot{z}_{t3d}
\end{bmatrix} = R_x \begin{bmatrix}
 \dot{x}_t \\
\dot{y}_t \\
\dot{z}_t
\end{bmatrix},$$

$$\begin{bmatrix}
 \ddot{x}_{t3d} \\
\ddot{y}_{t3d} \\
\ddot{z}_{t3d}
\end{bmatrix} = R_x \begin{bmatrix}
 \ddot{x}_t \\
\ddot{y}_t \\
\ddot{z}_t
\end{bmatrix}.$$
4.1.1 MATLAB-Simulink Model

Using the virtual target equations of motion discussed in the previous section of this chapter, 3D TS guidance model, DF model developed by Ferrin et al. [2] and Quadrotor UAV model described in the second chapter, the whole system was simulated in MATLAB. The following model shown in Fig. 4.1 was used to obtain the simulation results. The MATLAB’s inbuilt S-function and Simulink were used along with few user defined functions to simulate the whole system of UAV and controller.

![Fig. 4.1: MATLAB-Simulink model used to implement 3D TS](image)

4.2 Hardware Implementation

The hardware implementation was successfully achieved by using the RISC MoCap platform explained in Chapter-3 and ROS-PixHawk communication integration developed by Abhishek Manjunath [20]. A circular trajectory was chosen for comparison of control efforts commanded by TS and DF controllers. ROS-master was used on a Linux operating computer to setup the node to node communication for transferring date from one piece of code to another. For completing this loop, two computer operating the ubuntu-14.04 version of Linux were used. One computer is used as onboard computer to receive control commands, while other one serves as the ground station. Both computers use a secure Wi-Fi
network connection to exchange the information from one node to another. ROS-master runs on the ground station computer and everything running on the onboard computer is controlled from this machine by logging in through ROS-ssh [21] over a shared network. The states estimation, virtual target, and TS controller nodes run on the main ground computer, and only TS guidance controller commands are sent to the onboard computer. Once the ground station nodes are launched, the onboard computer’s PixHawk node is initiated to start sending TS controller commands to the autopilot.

4.2.1 Hardware Components

The general schematic of all the hardware components in loop is shown by Fig.4.2.

![Fig. 4.2: RISC MoCap RPTP hardware loop.](image-url)
CHAPTER 5
RESULTS

In this chapter, the results are obtained by applying different conditions to the target motion, vehicle controller. The results are classified into two sections for simulation and hardware implementation.

5.1 Simulation Results

Using the virtual target set of equations of motion in Chapter-4 in MATLAB simulation, the results were obtained for circular and figure-eight trajectories. The sampling time of 0.01 seconds was used to get higher resolution data.

5.1.1 Flat Circular Trajectory

The results for circular trajectory were obtained by using $\text{Radius} = 0.9m, R^* = 0.45m$, Vehicle speed $V_c = 0.37m/s$, trajectory altitude $n = 1m$, velocity controller gain $K_v = 3.5$. The plot for 3D trajectories for the virtual target, TS vehicle and DF vehicle have been shown in Fig.5.1. The control efforts commanded by 3D TS and DF controllers have been shown in Fig.5.2. The magnitude of velocities achieved by vehicle using TS and DF controllers while moving on the commanded trajectory have been shown in Fig.5.3. The positional errors generated by both vehicles while converging to the commanded path have been shown in Fig.5.4.

5.1.2 Inclined Figure-8 Trajectory

The results for figure-eight trajectory were obtained by using $\text{Radius} = 0.9m, R^* = 0.45m$, Vehicle speed $V_c = 0.37m/s$, trajectory altitude $n = 1m$, velocity controller gain $K_v = 3.5$. The plot for 3D trajectories for the virtual target, TS
Fig. 5.1: Circular trajectory plot from simulation results.

Fig. 5.2: Control efforts commanded by controllers.
Fig. 5.3: Magnitude of velocities achieved by TS and DF vehicles

Fig. 5.4: Positional error generated by both TS and DF vehicles
vehicle and DF vehicle have been shown in Fig. 5.5. The control efforts commanded by 3D TS and DF controllers have been shown in Fig. 5.6. The magnitude of velocities achieved by vehicle using TS and DF controllers while moving on the commanded trajectory have been shown in Fig. 5.7. The positional errors produced by both vehicles while converging to the commanded path have been shown in Fig. 5.8.

![3D Trajectories](image)

**Fig. 5.5:** Figure-eight trajectory plot from simulation results

### 5.2 Hardware Results

Using the virtual target set of equations of motion in Chapter-4, 3D TS model, and DF model, results were obtained for circular trajectory. The state estimation was run at 200Hz while the trajectory and controller nodes were run at only 100Hz. For getting the hardware results, low pass filters were used to filter the low-frequency noise in the vehicle states estimated from MoCap data. Low pass filters worked for few parameters, but few others had to be constrained by putting saturations. The virtual target controller uses the LOS distance
Fig. 5.6: Control efforts commanded by controllers

Fig. 5.7: Magnitude of velocities achieved by TS and DF vehicles
between vehicle to target to calculate target speed, and LOS distance is calculated by using vehicle and target states. Therefore the virtual target velocity computed from its controller had to be saturated within safe margins to ensure the smooth accelerations of the virtual target. The states of the vehicle were saturated to the physical constraints of the system. The position states were saturated to the limit of actually captured volume of MoCap system while rest of states were saturated based on the actual limits of Quadrotor UAV capabilities. For the further safety of the Quadrotor UAV, the output commanded accelerations for roll and pitch generated from both 3D TS and DF controllers were saturated to $0.2m/s^2$.

### 5.2.1 Flat Circular Trajectory

The results for circular trajectory were obtained by using $Radius = 0.9m, R^* = 0.45m$, Vehicle speed $V_c = 0.37m/s$, trajectory altitude $n = 1m$, velocity controller gain $K_v = 3.5$. The plot for 3D trajectories for the virtual target, TS vehicle and DF vehicle is shown in Fig.5.9. The control efforts commanded by 3D TS and DF controllers have been shown in
Fig. 5.10. The magnitude of velocities achieved by vehicle using TS and DF controllers while moving on the commanded trajectory have been shown in Fig. 5.11. The positional errors generated by both vehicles while converging to the commanded path have been shown in Fig. 5.4.

![3D Trajectories](image)

**Fig. 5.9:** Circular trajectory plot from hardware results.

### 5.2.2 Inclined Figure-8 Trajectory

For the figure-eight trajectory, the results were obtained by using semi-major axis to be $R_x = 1.35 \text{m}$ on X-axis, semi-minor axis to be $R_y = 0.9 \text{m}$ on Y-axis, second semi-minor axis to be $R_z = 0.3 \text{m}$ on Z-axis, $R^* = 0.45 \text{m}$, Vehicle speed $V_c = 0.3 \text{m/s}$, trajectory altitude $n = 1.3 \text{m}$, velocity controller gain $K_v = 3.5$. The result plots for 3D trajectories for the virtual target, TS vehicle and DF vehicle have been shown in Fig. 5.13. The control efforts commanded by 3D TS and DF controllers have been shown in Fig. 5.14. The magnitude of velocities achieved by vehicle using TS and DF controllers while moving on the commanded trajectory have been shown in Fig. 5.15. The positional errors generated by both vehicles while converging to the commanded path have been shown in Fig. 5.16.
Fig. 5.10: Control efforts commanded by controllers

![Control Effort Graph]

Fig. 5.11: Magnitude of velocities achieved by vehicle using TS and DF controllers

![Velocity Graph]

Table 5.1: Root Mean Square (RMS) of position error for simulation data

<table>
<thead>
<tr>
<th>Trajectory</th>
<th>TS</th>
<th>DF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circle</td>
<td>0.0216 m</td>
<td>0.027 m</td>
</tr>
<tr>
<td>Figure-8</td>
<td>0.0609 m</td>
<td>0.061 m</td>
</tr>
</tbody>
</table>
Fig. 5.12: Positional error generated by vehicle using TS and DF controllers.

Fig. 5.13: Figure-eight trajectory plot from simulation results.

Table 5.2: Root Mean Square (RMS) of position error for hardware test data

<table>
<thead>
<tr>
<th>Trajectory</th>
<th>TS</th>
<th>DF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circle</td>
<td>0.1892 m</td>
<td>0.1227 m</td>
</tr>
<tr>
<td>Figure-8</td>
<td>0.1662 m</td>
<td>0.062 m</td>
</tr>
</tbody>
</table>
Fig. 5.14: Control efforts commanded by controllers

Fig. 5.15: Magnitude of velocities achieved by TS and DF vehicles
Fig. 5.16: Positional error generated by both TS and DF vehicles.

Table 5.3: Root Mean Square (RMS) of position error for hardware test data after convergence of vehicle path to target path

<table>
<thead>
<tr>
<th>Trajectory</th>
<th>TS</th>
<th>DF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circle</td>
<td>0.0252 m</td>
<td>0.0425 m</td>
</tr>
<tr>
<td>Figure-8</td>
<td>0.1313 m</td>
<td>0.0666 m</td>
</tr>
</tbody>
</table>
CHAPTER 6
CONCLUSION

The concept of TS was successfully developed from 2D to 3D and confirmed for its stability by simulation and hardware results. RISC MoCap RPTP platform designed and constructed for this research worked successfully, hence was used for hardware implementation and results. Despite some difficulties of hardware implementation due to real world non-linearities and system noise, positive results were obtained.

Simulation and hardware results were obtained to compare the performance of TS and DF controllers by using same trajectory and vehicle parameters. Results in Fig.5.1, Fig.5.5, Fig.5.9 and, Fig.5.13 confirm the main purpose of TS guidance to shape the vehicle’s trajectory for its smooth entrance to target’s trajectory. The convergence of vehicle trajectory to the target trajectory can easily be noticed from the root mean square data provided in Table.5.1, Table.5.2 and Table.5.3 for positional errors. From the trajectory figures, it can be seen that TS vehicle tries to enter the target trajectory smoothly, while DF vehicle is attempting to run towards the target to minimize the difference between vehicle’s and target’s positions and velocities. This fact shows that in TS guidance, instead of forcing itself to be on the target waypoint, UAV rather utilizes its control effort to be on the target trajectory at any instant of time. The vehicle using TS guidance for path following overshoots lesser than that of DF while merging to the commanded trajectory. These facts pointed towards the less wastage of onboard available energy and increased flight time of Quadrotor UAV.

The velocity plots given by Fig.5.3, Fig.5.7, Fig.5.11 and Fig.5.15 shows that in the case of TS controller, the vehicle converges to the commanded velocity on the given trajectory quicker with smaller overshoots in magnitude than that of DF.

6.1 Future Work
TS and DF controllers can be mixed and mutually switched during an autonomous flight
to always maintain the vehicle’s convergence to target’s aggressive and random trajectory. In this way, the commanded control effort can further be optimized by switching to DF controller whenever path becomes extremely aggressive. Otherwise, the vehicle may diverge from the path and take a long time and more energy to converge back to target’s path if using TS controller. TS guidance can be switched back into control whenever a relative smoother path is encountered.
REFERENCES


