AN OPTIMUM VISION-BASED CONTROL OF ROTORCRAFTS
CASE STUDIES: 2-DOF HELICOPTER & 6-DOF QUADROTOR

A Thesis
Submitted to the Faculty of Graduate Studies and Research
In Partial Fulfillment of the Requirements
for the Degree of
Master of Applied Science
in Electronic Systems Engineering
University of Regina

by
Maryam Alizadeh
Regina, Saskatchewan, Canada
July, 2013

Copyright © 2013: M. Alizadeh
Maryam Alizadeh, candidate for the degree of Master of Applied Science in Electronic Systems Engineering, has presented a thesis titled, *An Optimum Vision-Based Control of Rotorcrafts Case Studies: 2-DOF Helicopter & 6-DOF Quadrotor*, in an oral examination held on July 29, 2013. The following committee members have found the thesis acceptable in form and content, and that the candidate demonstrated satisfactory knowledge of the subject material.

External Examiner: Dr. Nader Mobed, Department of Physics

Supervisor: Dr. Raman Paranjape, Electronic Systems Engineering

Committee Member: *Dr. Mehran Mehrandezh, Electronic Systems Engineering

Committee Member: Dr. Paul Laforge, Electronic Systems Engineering

Chair of Defense: Dr. Hussameldin Ibrahim, Industrial Systems Engineering

*Participated via Teleconference*
ABSTRACT

An unmanned aerial vehicle (UAV) is an aircraft capable of sustained flight without a human operator on board which can be controlled either autonomously or remotely (e.g., by a pilot on the ground). In recent years, the unique capabilities of UAVs have attracted a great deal of attention for both civil and military application. UAVs can be controlled remotely by a crew miles away or by a pilot in the vicinity. Vision-based control (also called visual servoing) refers to the technique that uses visual sensory feedback information to control the motion of a device. Advancements in fast image acquisition/processing tools have made vision-based control a powerful UAV control technique for various applications. This thesis aims to develop a vision-based control technique for two sample experimental platforms, including: (1) a 2-DOF (degrees of freedom) model helicopter and (2) a 6-DOF quadrotor (i.e. AR.Drone), and to characterize and analyze response of the system to the developed algorithms.

For the case of 2-DOF, the behavior of the model helicopter is characterized and the response of the system to the control algorithm and image processing parameters are investigated. In this section of experiments, the key parameters (e.g., error clamping gain and image acquisition rate) are recognized and their effect on the model helicopter behavior is described. A simulator is also designed and developed in order to simplify working with the model helicopter. This simulator enables us to conduct a broad variety of tests with no concerns about the hardware failure or experimental limitations. It also can be used as a training tool for those who are not familiar with the device and can make
them ready for real-world experiments. The accuracy of the designed simulator is verified by comparing the results of real tests and simulated ones.

A quintic polynomial trajectory planning algorithm is also developed using the aforementioned simulator so that servoing and tracking the moving object can be achieved in an optimal time. This prediction, planning and execution algorithm provides us with a feasible trajectory by considering all of the restrictions of the device. The necessity of re-planning is also addressed and all of the involved factors affecting operation of the algorithm are discussed.

The vision-based control structure developed for the 6-DOF quadcopter provides the capability for fully autonomous flights including takeoff, landing, hovering and maneuvering. The objective is to servo and track an object while all 6 degrees of freedom are controlled by the vision-based controller. After taking off, the quadcopter searches for the object and hovers in the desired pose (position and direction) relative to that. In the case that the object cannot be found, the quadcopter will land automatically. A motion tracking system consists of a set of infrared cameras (i.e. OptiTrack system) mounted in the experiment environment, which is used to provide the accurate pose information of the markers on the quadcopter. By comparing the 3D position and direction of the AR.Drone relative to the object obtained by the vision-based structure and the information provided by the OptiTrack, the results of the developed algorithm are evaluated. Results of developed algorithms in this section provide a flexible and robust vision-based fully-autonomous controlled aerial platform for hovering, maneuvering, servoing and tracking in small-size lab environments.
ACKNOWLEDGEMENTS

I wish to express my most sincere gratitude to my supervisor, Dr. Raman Paranjape, for his continuous support, insightful guidance and invaluable advice throughout the present study.

My deepest appreciation goes to Dr. Mehran Mehrandezh, for his kind helps and advices during the course of my research work.

I would also like to take this opportunity to thank the members of my thesis committee for their dedication to reviewing of this thesis and their constructive comments.

I am grateful to TRTech (former TRLabs) Research Consortium, NSERC - Natural Sciences and Engineering Research Council of Canada, and Faculty of Graduate Studies and Research at the University of Regina for their financial supports.

I also thank Mr. Mehrdad Bakhtiari, Mr. Seang cau, Mr. Chayatat Ratanasawanya, Mr. Zhanle Wang and other members of UAV lab at the University of Regina for their supports and helps. Finally, I would like to thank all who was important to the successful realization of this thesis, as well as expressing my apology to those who I could not mention their name one by one.
This thesis is dedicated to my husband, Ahmad, and my parents; All I have accomplished were only possible due to their love, encouragement and support.
TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT .................................................................</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS ...........................................................</td>
</tr>
<tr>
<td>DEDICATIONS .................................................................</td>
</tr>
<tr>
<td>TABLE OF CONTENTS ............................................................</td>
</tr>
<tr>
<td>TABLE OF FIGURES .............................................................</td>
</tr>
<tr>
<td>CHAPTER 1 INTRODUCTION .................................................................</td>
</tr>
<tr>
<td>1.1. Background ..................................................................</td>
</tr>
<tr>
<td>1.1.1. Visual servoing structure ........................................</td>
</tr>
<tr>
<td>1.2. Literature Review ........................................................</td>
</tr>
<tr>
<td>1.2.1. 2-DOF model helicopters ........................................</td>
</tr>
<tr>
<td>1.2.2. Quadrotors ............................................................</td>
</tr>
<tr>
<td>1.3. Experimental Platform ...................................................</td>
</tr>
<tr>
<td>1.3.1. 2-DOF Model Helicopter ...........................................</td>
</tr>
<tr>
<td>1.3.2. 6-DOF Model Helicopter ...........................................</td>
</tr>
<tr>
<td>1.4. Research objectives ......................................................</td>
</tr>
<tr>
<td>1.5. Contributions ................................................................</td>
</tr>
<tr>
<td>1.6. Thesis structure ..........................................................</td>
</tr>
<tr>
<td>CHAPTER 2 SYSTEM DYNAMICS &amp; CONTROL ALGORITHM FOR 2-DOF ROTORCRAFT .................................................................</td>
</tr>
<tr>
<td>2.1. Mathematical Modeling and Control Algorithm .......................</td>
</tr>
<tr>
<td>2.1.1. Joint-level LQR controller ..........................................</td>
</tr>
<tr>
<td>2.2. Image processing ..........................................................</td>
</tr>
<tr>
<td>2.3. Target Depth Estimation ..................................................</td>
</tr>
<tr>
<td>2.4. Incremental motion calculation ........................................</td>
</tr>
<tr>
<td>2.5. Characterization and Sensitivity Analysis of Effective Parameters</td>
</tr>
<tr>
<td>2.5.1. Error-Clamping-Gain and Image Acquisition Rate ...............</td>
</tr>
<tr>
<td>2.5.2. Experimental Setup .....................................................</td>
</tr>
<tr>
<td>2.6. Results and discussion ....................................................</td>
</tr>
<tr>
<td>2.6.1. Error clamping gain(ECG) and Image acquisition rate(FPS)</td>
</tr>
<tr>
<td>2.6.2. Initial position of the object ........................................</td>
</tr>
<tr>
<td>2.6.3. Location of the Camera ...............................................</td>
</tr>
<tr>
<td>2.7. Proportional-Derivative Controller ......................................</td>
</tr>
</tbody>
</table>
2.8. Summary and conclusion.......................................................... 38

CHAPTER 3 SIMULATOR & TRAJECTORY PLANNING FOR 2-DOF ROTORCRAFT .......... 40

3.1. Simulator .............................................................................. 40
   3.1.1. Simulating the Attached camera ..................................... 42
   3.1.2. Sampling rate compatibility ........................................... 45
   3.1.3. Evaluation of the Simulator ............................................ 47
      3.1.3.1. Proportional Controller (effect of ECG) .................. 48
      3.1.3.2. Proportional-Derivative Controller ......................... 49

3.2. Trajectory planning................................................................. 50
   3.2.1. Trajectory Planning Algorithm ...................................... 53
   3.2.2. Optimal Trajectory Planning ......................................... 54

3.3. Trajectory planning results and discussion ................................ 55
   3.3.1. Set 1: Trajectory Planning (with no re-planning) ............ 56
   3.3.2. Set 2: Trajectory Planning (with re-planning) ............... 59
   3.3.3. Set 3: Trajectory Planning with different values for the controller gain 62

3.4. Summary and Conclusions...................................................... 64

CHAPTER 4 6-DOF ROTORCRAFT OVERVIEW: DYNAMIC & CONTROL ALGORITHM ....... 67

4.1. Introduction .......................................................................... 67

4.2. The Parrot AR.Drone Platform.............................................. 69
   4.2.1. Aerial Vehicle ................................................................. 69
   4.2.2. On-Board Electronics .................................................... 71

4.3. AR.Drone Start-up................................................................. 72

4.4. Vision and Inertial Navigation Algorithms .............................. 73
   4.4.1. Vision Algorithm ........................................................... 73
   4.4.2. Inertial Sensor Calibration ............................................. 74
   4.4.3. Attitude Estimation ....................................................... 75
   4.4.4. Inertial sensors usage for video processing .................... 76

4.5. Aerodynamics Model For Velocity Estimation ......................... 76

4.6. Control Structures ............................................................... 78

4.7. Summary and Conclusion...................................................... 81

CHAPTER 5 VISION BASED CONTROL OF 6-DOF ROTORCRAFT ................................ 82

5.1. Lab Space Setup ................................................................... 82

5.2. Vision-based Control Algorithm .......................................... 84
5.2.1. Image processing algorithm ......................................................... 85
5.2.2. Control Algorithm ........................................................................ 87
  5.2.2.1. Pitch (longitudinal) motion control ........................................ 89
  5.2.2.2. Lateral motion control .......................................................... 91
  5.2.2.3. Vertical motion Control ......................................................... 92
  5.2.2.4. Gain range calculation .......................................................... 93
5.3. Design of Tests .................................................................................. 93
5.4. Results and discussion ...................................................................... 94
5.5. Summary and conclusion .................................................................. 101

CHAPTER 6 CONCLUSION AND FUTURE WORKS ................................................. 102
  6.1. Future Work ................................................................................... 104

REFERENCES ............................................................................................ 105
# TABLE OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Visual servoing structure</td>
<td>3</td>
</tr>
<tr>
<td>1.2</td>
<td>The 2-DOF model helicopter used in experiments</td>
<td>8</td>
</tr>
<tr>
<td>1.3</td>
<td>The proposed vision-based control system for a 6-DOF quadrotor (i.e., AR.Drone). Photo is taken from <a href="http://www.bit-tech.net">http://www.bit-tech.net</a></td>
<td>9</td>
</tr>
<tr>
<td>2.1</td>
<td>2-DOF model helicopter</td>
<td>17</td>
</tr>
<tr>
<td>2.2</td>
<td>2-DOF model helicopter dynamics model. [35]</td>
<td>18</td>
</tr>
<tr>
<td>2.3</td>
<td>Simulink image processing algorithm</td>
<td>22</td>
</tr>
<tr>
<td>2.4</td>
<td>The ping-pong ball’s (a) acquired image, (b) segmented location, and (c) acquired image with desired and current location superimposed</td>
<td>23</td>
</tr>
<tr>
<td>2.5</td>
<td>Top view of the object (ball) and its projection on image plane</td>
<td>24</td>
</tr>
<tr>
<td>2.6</td>
<td>Diagram of motion of the ball from location #1 to Location #2</td>
<td>25</td>
</tr>
<tr>
<td>2.7</td>
<td>Experimental setup for evaluating the developed vision-based algorithm</td>
<td>30</td>
</tr>
<tr>
<td>2.8</td>
<td>Variation of the closed-loop poles versus the sampling rate and the error-clamping gain</td>
<td>31</td>
</tr>
<tr>
<td>2.9</td>
<td>Variation of closed-loop poles versus ECG, in the S-domain</td>
<td>32</td>
</tr>
<tr>
<td>2.10</td>
<td>Variation of closed-loop poles versus for different initial positions</td>
<td>33</td>
</tr>
<tr>
<td>2.11</td>
<td>Three selected location for attaching the camera</td>
<td>34</td>
</tr>
<tr>
<td>2.12</td>
<td>Radial centroid error, $E_c$, versus Time for (a) camera location #1, (b) camera location #2 position, (c) camera location #3</td>
<td>35</td>
</tr>
<tr>
<td>2.13</td>
<td>Vision-based control algorithm (Proportional Controller)</td>
<td>36</td>
</tr>
<tr>
<td>2.14</td>
<td>Vision-based control algorithm (Proportional-Derivative Controller)</td>
<td>37</td>
</tr>
<tr>
<td>2.15</td>
<td>Variation of closed-loop poles versus Derivative controller gain ($K_d$), in S-domain</td>
<td>38</td>
</tr>
<tr>
<td>3.1</td>
<td>Real-World vision-based control algorithm</td>
<td>41</td>
</tr>
<tr>
<td>3.2</td>
<td>World (W) and Camera (C) frames, axes and sign convention</td>
<td>43</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>3.3</td>
<td>Location of ‘Rate Transition’ blocks in the...</td>
<td></td>
</tr>
<tr>
<td>3.4</td>
<td>Variation of closed-loop poles of the simulated and real system versus ECG value, in S domain...</td>
<td></td>
</tr>
<tr>
<td>3.5</td>
<td>Radial centroid error (Ec) versus time plotted by varying the Kd parameter within [0.01, 0.1] interval while ECG=0.</td>
<td></td>
</tr>
<tr>
<td>3.6</td>
<td>Trajectory planning algorithm block diagram...</td>
<td></td>
</tr>
<tr>
<td>3.7</td>
<td>System’s response to planned trajectories: (a – f) Pitch/yaw trajectories, (g – i) control inputs in form of DC motor applied voltages, (j-o) velocity and acceleration profiles.</td>
<td></td>
</tr>
<tr>
<td>3.8</td>
<td>System’s response to re-planned trajectories: (a – f) Pitch/yaw trajectories, (g – i) control inputs in form of DC motor applied voltages (j-o) velocity and acceleration profiles.</td>
<td></td>
</tr>
<tr>
<td>3.9</td>
<td>The effect of the control gains on the overall performance of the system with no re-planning, (a-f) pitch and yaw responses (g-i) motor voltages, (j-o) velocity/acceleration profile.</td>
<td></td>
</tr>
<tr>
<td>4.1</td>
<td>AR.Drone hull for (a) indoor applications and (b) outdoor applications. [44]</td>
<td></td>
</tr>
<tr>
<td>4.2</td>
<td>Schematic of quadrotor maneuvering in (a) pitch direction, (b) roll direction, and (c) yaw direction.</td>
<td></td>
</tr>
<tr>
<td>4.3</td>
<td>An example of velocity estimation: vision-based (blue), aerodynamics model (green), and fused (red) velocity estimates. [45]</td>
<td></td>
</tr>
<tr>
<td>4.4</td>
<td>A snapshot of the AR.Drone's graphical user interface.</td>
<td></td>
</tr>
<tr>
<td>4.5</td>
<td>State machine description. [45]</td>
<td></td>
</tr>
<tr>
<td>4.6</td>
<td>Data fusion and control architecture. [45]</td>
<td></td>
</tr>
<tr>
<td>5.1</td>
<td>Schematic of the experimental environment for the 6-dof helicopter of this study.</td>
<td></td>
</tr>
<tr>
<td>5.2</td>
<td>(a) Location of IR markers on the AR.Drone, (b) The triangular shaped object defined by the markers.</td>
<td></td>
</tr>
<tr>
<td>5.3</td>
<td>Developed image processing algorithm.</td>
<td></td>
</tr>
</tbody>
</table>
Figure 5.4: (a) Captured image of the object and (b) Computed binary image after segmentation, the calculated centroid point is shown as the black dot. 87
Figure 5.5: The control algorithm for manually controlled AR.Drone by the Pilot. 88
Figure 5.6: Autonomous control of pitch/longitudinal motion. 89
Figure 5.7: Diagram of the ball, current and desired positions and resulted lateral error, EX. 91
Figure 5.8: Diagram of the ball, current and desired positions and resulted lateral error, EX. 92
Figure 5.9: Autonomous control of vertical motion algorithm. 93
Figure 5.10: Generated commands by the vision-based control algorithm for (a) Roll, (b) Pitch and (c) vertical speed. 95
Figure 5.11: Recorded error values - achieved from visual information- versus time; (a) Horizontal error value (Ex), (b) Vertical error value (Ey), (a) Distance error value (ED). 97
Figure 5.12: Comparison between normalized error values obtained from vision-based algorithm and OptiTrack system; (a) Normalized horizontal error, (b) Normalized vertical error, (c) Normalized distance error. 100
CHAPTER 1
INTRODUCTION

An unmanned aerial vehicle (UAV) is an aircraft capable of sustained flight without a human operator on board which can be controlled either autonomously or remotely (e.g., by a pilot on the ground). UAVs are predominantly used for military applications; however in recent years the unique capabilities of UAVs have attracted a great deal of attention for civil applications such as rescue missions [1], remote sensing [2], transport monitoring [3], military missions [4], commercial aerial surveillance [5], oil, gas and mineral exploration [6], forest fire detection [7], and scientific research. UAVs can generally be classified into two categories, fixed-wing and rotary-wing (rotorcraft) UAVs. Rotorcrafts (e.g., helicopters) are more common for surveillance applications due to their capabilities of vertical take-off, hovering and landing in small and relatively rugged areas.

This thesis aims to develop an optimum vision-based control algorithm for two experimental UAV platforms. The first platform is a 2-DOF (degree of freedom) model helicopter - equipped with two rotors in the front and rear - and the other platform is a 6-DOF quadrotor - which is equipped with four rotors. The purpose of the developed vision-based algorithms is to control all degrees of freedom of the rotorcrafts to provide a fully autonomous flight.

1.1. Background

Vision-based control (also called visual servoing or visual servo-control) refers to the techniques that use visual sensory feedback information to control the motion of a
Visual servoing techniques can be categorized into the following three types:

- **Image-based visual servoing (IBVS)**[8]: This technique is based on the error between current and desired features on the image plane (e.g., coordinates of visual features, lines or moments of regions). This study uses image-based visual servoing techniques.

- **Position-based visual servoing (PBVS)** [9]: In this technique, the pose of the object of interest is estimated with respect to the camera, and then a command is issued to the controller. Unlike the image-based technique, this technique uses extracted image features to estimate 3D position information.

- **Hybrid techniques**: This involves using a combination of image-based and position based techniques.

1.1.1. **Visual servoing structure**

The goal of visual servoing is to minimize an error value $e(t)$ given by [10]:

$$\text{Minimize } \{e(t), \ t\}; \quad e(t) = s(t) - s_d$$  \hspace{1cm} (1.1)

where $t$ is time, and $s(t)$ and $s_d$ are current and desired visual feature vectors, respectively (Figure 1.1). For image based control, the vector $s$ consists of a set of features that are immediately available in the image data (e.g., coordinates or lines) and the vector is defined in image space, $^i s$. For position-based control, the vector $s$ consists of a set of 3-D parameters and the vector is defined in Cartesian space, $^c s$. 
The servoing schemes can be further divided into two structures, depending on whether the control structure is hierarchical [11]. If the joint-level inputs are directly generated by the system, it is a direct visual servo structure, and if the system only provides the set-points (as the inputs) for the joint-level controller, the system is a hierarchical vision-based control structure. Another type of classification of visual servoing techniques is based on the components of a servoing system. For instance, based on the location of the camera, the configuration can be eye-in-hand or hand-eye. The following briefly summarizes the main features of image and position based control algorithm [9].

- In image-based algorithms, image Jacobian calculation is required, while for position-based algorithms, pose estimation is needed.

- Local stability is guaranteed by an image-based approach, while position-based method guarantees global stability.

- Position-based algorithms are the better alternative when the camera is far from the object.
The image-based method provides optimal feature point movement in image- and position-based approach results in optimal camera movements in a 3D coordinate system.

In position-based algorithms, accurate camera calibration and a 3D model of the object are required.

Pose estimation is one important phase of position-based servoing algorithms which uses the 2D imagery data and determines the 3D position information. Three classes of pose estimation methodologies can be distinguished including (1) Analytic or geometric methods; (2) Genetic algorithm; and (3) Learning-based methods. The pose estimation problem (also referred to as a Perspective-n-Point, or PnP problem) is aimed to be solved for the position and orientation of a camera, given its intrinsic parameters (e.g. focal length, principal point) and a set of n correspondences between 3D points and their 2D (two-dimensional) projections on the image (i.e. given a set of n corresponding points, what is the pose of the object in 3D space) [12]. There are mainly two categories of algorithms for solving this problem: iterative and non-iterative methods [13].

1.2. Literature Review

Following addresses the previous research work on different degree-of-freedom rotorcrafts, including those used in this study.

1.2.1. 2-DOF model helicopters

Within the past two decades, controlling unmanned aerial vehicles (UAVs) has been an active research topic. Kaloust et al. [14] proposed a non-linear Lyapunov-based control for take-off and landing of helicopters, claiming that the semi-global stability is guaranteed. Later, a non-linear predictive control for pitch and yaw movements was proposed in [15]. Since this algorithm was based on state-space Generalized Predictive
Control (GPC) law, a better attraction zone was achieved in comparison to linear controllers. In [16], a combination of the optimal Linear Quadratic Regulator (LQR) and the Sliding Mode Control (SMC) was developed. Simulation and experimental results of this controller on a helicopter model capable of pitch and yaw motion illustrated an optimal control performance and its robustness against external disturbances. In 2008, an intelligent control inspired by an emotional model of human brain was presented [17]. This Brain Emotional Learning Based Intelligent Controller (BELBIC) was simulated on a nonlinear model of a 2DOF helicopter. In [18], performance of a blindfolded human operator to control a 2DOF model helicopter was investigated. In this study, the operator navigates the model helicopter to the zone of interest using body-referenced vibro-tactile sensory information. The end-of-motion precise maneuvers were performed by an LQR controller.

Recent developments in imaging sensors and image processing power have provided an exclusive opportunity for vision-based control of UAVs. An investigation on autonomous control of quadrotor helicopters equipped with a monocular vision system has been done in [19]. By using vision information, the position and attitude of the vehicle were estimated and fed back to controller in order to stabilize the helicopter. In [20], in addition to a single camera, GPS and an Inertial Measurement Unit (IMU) have been used. These sensors provide 6-DOF information for the controller of a twin-rotor helicopter to track features in an urban area.

1.2.2. Quadrotors

Quadrotors are categorized as rotorcraft (as opposed to fixed-wing aircraft). Quadrotors generally use symmetrically pitched blades, and their control is achieved by changing the pitch and/or rotation rate of the rotors (resulting in altering torque load and
thrust/lift characteristics). Quadrotor configurations are considered as a solution to the torque-induced control issues (the common problems in vertical flight). In recent years, quadrotor designs have become popular in UAV research works [21], [22].

Quadcopters have several advantages over other rotary wing UAVs such as helicopters. They do not require mechanical linkages to vary the rotor blade pitch angle as they spin. In other words, their control actuation consists of changing motor speeds rather than changing blade pitch [23]. Furthermore, the use of four rotors allows each individual rotor to have a smaller size for small-scale UAVs; this makes the device safer for close operation [6]. Due to their construction and control simplicity, quadrotors are frequently used in research projects [24]. Some recent quadrotors include: Bell Boeing Quad TiltRotor, Aermatica Spa's Anteos [25], AeroQuad, ArduCopter [26], OpenPilot and Parrot AR.Drone [27]. Such advantages have attracted a great deal of interest in the field of vision-based control.

Visual information required for vision-based control of UAVs can be provided by a ground-based camera in an eye-hand configuration [28], or the camera can be on-board, providing imagery information while mounted on the quadrotor [29], [30]. In addition, multi cameras can also be employed, as in [31] which used two cameras as a “visual odometer” for helicopter position estimation and also in [32] two cameras are used in a hybrid configuration (one ground based camera and one on-board camera) for pose estimation purposes. Beside pose estimation, many other applications for vision-based controlled quadrotors have been subjects of several recent research works.

An algorithm was introduced in [27] to use the AR.Drone as an external navigation system for a formation of mobile robots. A 3D model-based tracking scheme was also introduced for indoor position control of quadrotors [33]. This algorithm
requires a pre-determined 3D model of the flight area and also the inertial sensory information to locate itself and control the position. Despite numerous research works on vision-based control of quadrotors, developing a robust and efficient, yet accurate algorithm that does not require pre-defined information (e.g., flight condition and environment) is still a major challenge and needs further research efforts.

1.3. Experimental Platform

This thesis is based on two experimental platforms including: (1) a 2-DOF (degrees of freedom) model helicopter, and (2) a 6-DOF quadrotor (AR.Drone), which are used as the case studies for the purpose of vision-based control. This section provides an overview of these two cases.

1.3.1. 2-DOF Model Helicopter

The first experimental platform of this study is a 2-DOF model helicopter by QUANSER INC. [35]. The platform is presented in Figure 1.2. Two propellers are driven using DC motors mounted at the two ends of a rectangular frame, and make the device capable of rotation in pitch and yaw directions. A webcam is attached to the back to transfer the visual information to the device, giving it the potential of tracking a randomly moving object (here, a ping-pong ball).

The horizontal and vertical propellers control the helicopter nose elevation and side to side movement around the pitch and yaw axes, respectively. The high-resolution encoders are employed to measure the pitch and yaw angles. The pitch encoder outputs 4096 counts per revolution in quadrature mode, and the yaw encoder outputs 8192 counts per revolution. The helicopter has the ability to rotate indefinitely in the yaw direction,
while the pitch-wise motion is limited to -40.5 to 40.5 degrees. The maximum voltages supplied to the pitch and yaw motors are ±24 V and ±15 V, respectively.

Figure 1.2: The 2-DOF model helicopter used in experiments.

1.3.2. 6-DOF Model Helicopter

The second experimental platform of this research work is a 6-DOF quadrotor rotorcraft, used for the purpose of vision-based control (Figure 1.3). The AR.Drone, a remotely controlled flying quadrotor built by the French company Parrot, was selected for this purpose. The AR Drone is expected to be a popular research and educational platform because of its stability, affordability, sensory equipment and open API. It has already been used for several visual-based control experiments (e.g., [36], [37], and [38]).

The AR.Drone consists of a carbon-fiber skeleton on which four 15 W electric motors, powered by a rechargeable (11 V) lithium battery pack, are mounted to drive the propellers. Two cameras are fitted to the device, a wide-angle VGA camera at the front and a high speed vertical camera. The AR Drone is equipped with several motion sensors, providing pitch, roll, and yaw measures, which are used for automatic stabilization. AR.Drone employs a Linux-based embedded microcontroller which communicates with
the operator through Wi-Fi and universal serial bus (USB). The vision-based control proposed for this quadrotor is aimed to use the imagery information, making the quadrotor capable of automatically locating a target object and hover in front of it. Real-time images acquired by the front camera are used for this purpose.

Figure 1.3: The proposed vision-based control system for a 6-DOF quadrotor (i.e., AR.Drone). Photo is taken from http://www.bit-tech.net.
1.4. Research objectives

The overall objective of this research work is to develop an optimum vision based control algorithm for different aerial platforms. The detailed objectives are categorized based on the two selected research platforms of this study, the 2-DOF model helicopter and the 6-DOF quadrotor (AR.Drone), as outlined in the following.

2-DOF model research platform:

- Characterizing the behavior of the system in response to developed vision-based control for the 2-DOF model helicopter

- Optimizing the effective parameter values so that the control algorithm can be adjusted to any flight conditions

- Proposing a simulator as an evaluating tool for the developed vision-based control algorithm. This simulator helps to examine the possible alternatives in controlling the model helicopter before implementing them on the real model.

- Developing a non-linear polynomial trajectory planning algorithm in order to optimize the travelled trajectory by the helicopter in servoing and tracking modes. The algorithm plans a trajectory for the 2-DOF model helicopter from the current to the desired locations. The planned trajectory shall be permissible and none of the constraints be violated, therefore the dynamics of the system and also all of the involved constrains and limitations are considered in trajectory planning procedure.
6-DOF model research platform:

- Developing a vision-based fully autonomous control algorithm for a 6-DOF quadrotor capable of servoing an object in a confined area, without any required pre-defined information about the flight environment and condition.

- Proposing an image processing algorithm that can recognize the object in the provided image and extract the image point and size of the object required for calculating the navigation commands.

- Introducing a controller which generates high-level navigation commands using the provided visual information and directs them to the quadrotor through a Wi-Fi connection.

- Validation of the results to examine the behavior of the quadrotor and ensure the accuracy and reliability of the developed algorithm.
1.5. Contributions

Two optimum vision-based control structures are proposed and applied on two research platforms, 2-DOF model helicopter and 6-DOF quadrotor (AR.Drone). In followings, the main accomplishments achieved during this study have been summarized.

2-DOF model research platform:

- A vision-based control algorithm – capable of servoing and tracking an object – is optimized so that the 2DOF model helicopter can be adapted to any flight conditions and environment. The behavior of system is characterized in response to the possible effective parameters, and their optimum values for every flight circumstances are also presented.

- A Proportional-Derivative (PD) controller is enhanced from the previously developed controller in order to improve the system’s behavior in terms of stability and pace of the response. The provided results confirm that the newly introduced structure has successfully increased stability of the system without reducing the reaction speed.

- A simulator has been developed as an evaluation tool that can be employed to study all possible alternatives before implementation in the real-world system. The simulated experiments’ results certify the compatibility of the simulator and the real-world structure. The correspondence between the simulator and the real system makes the simulator a powerful training tool that can be used for educating beginners before operating the real system.

- The simulator has been engaged in developing a non-linear trajectory planning algorithm for servoing and tracking purposes. The planned trajectory not only provides the position commands, step by step, but also directs the velocity and acceleration orders for every step during the flight. In this regard, a quintic polynomial trajectory is selected to ensure the continuity of the velocity and acceleration profiles during the flight period.
• Permissibility of planned trajectory is guaranteed due to primary considerations and assumptions that have been made based on the dynamics of the system and the existing limitations. In tracking mode, where the object is moving, or in the case that the controller is not able to follow the planned trajectory, the trajectory shall be re-planned. Therefore, the necessity of re-planning is checked frequently and the re-planning algorithm will be activated when it is required.

6-DOF quadrotor research platform (AR.Drone)

• A vision-based control algorithm is developed for a 6-DOF quadrotor, which enables it to autonomously fly, recognize the object and servo it without any pre-defined information requirements about the flight circumstances. This research work provides a robust mechanism for fully autonomous visual servoing using a 6-DOF rotorcraft.

• A powerful real-time image processing scheme is developed as the visual information provider that can identify the object in the captured image and provide the required information about the object's location and size. In the object recognition procedure, the captured color image is converted to a HSV space image to ease identification and segmentation of the object.

• The computed visual information is used by a command generator part consisting of a PID controller, which creates the navigation commands and directs them to the drone. This algorithm is able to communicate with the rotorcraft through a Wi-Fi connection, transmit the control commands and receive the visual information provided by the on-board camera.

• The reliability and accuracy of the developed structure is validated by comparing the obtained results with the measurements from an external motion tracking system (OptiTrack system).

1.6. Thesis structure

This thesis is divided to two major parts:
Part I focuses on vision-based control of a 2-DOF rotorcraft. In this part, Chapter 2 addresses the system dynamic and the control algorithm; thereafter, the developed algorithm is characterized with respect to parameters that can affect the behavior of the system. A Proportional-Derivative (PD) controller is also introduced to improve the previously developed algorithm. Chapter 3 proposes a simulator (as an examination tool) for reproducing the behavior of the real system. Furthermore, a new trajectory planning algorithm, developed in this study, will be presented in this chapter.

Part II focuses on vision-based control of a 6-DOF rotorcraft. In this part, Chapter 4 addresses the system dynamic and the control algorithm of the 6-DOF quadrotor research platform; and Chapter 5 presents the proposed vision-based structure for fully autonomous control of the rotorcraft. The experimental results and their evaluation are also provided and discussed. Chapter 6 concludes the thesis work with possible extensions.
PART I:

VISION-BASED CONTROL OF A

2-DOF ROTORCRAFT
CHAPTER 2
SYSTEM DYNAMICS & CONTROL ALGORITHM FOR 2-DOF ROTORCRAFT

This chapter presents the system dynamics and the development of a vision-based control algorithm for a 2-DOF model rotorcraft capable of servoing and tracking an arbitrary moving object by rotating in pitch and yaw dimensions. At first the mathematical model of the helicopter and its control algorithm is explained. Thereafter, the components of the proposed control structure are described. At the next step, system characterization with respect to effective parameters is presented. Finally, an improved PD controller is proposed and evaluated.

2.1. Mathematical Modeling and Control Algorithm

The 2-DOF model helicopter used for this study, manufactured by Quanser Inc.¹ [35], consists of a model helicopter installed on a fixed base with two propellers driven by a couple of DC motors (Figure 2.1). The horizontal (front) and vertical (back) propellers control the helicopter nose elevation and side to side movement by generating the pitch and yaw angles, respectively. Two high-resolution encoders are employed to measure the pitch and yaw angles. The pitch encoder outputs 4096 counts per revolution while the yaw encoder outputs 8192 counts per revolution. The helicopter has the ability to rotate indefinitely in the yaw direction while the pitch-wise motion is limited to ±40.5 degrees. Pitch angle increases positively when the helicopter nose is moving upward and the yaw angle is defined as positive when the helicopter’s nose rotates in a clock-wise

¹ www.quanser.com
direction. The maximum voltages, supplied to the pitch and yaw motors, are ± 24 V and ± 15 V, respectively.

Figure 2.1: 2-DOF model helicopter

2.1.1. Joint-level LQR controller

A joint-level model-based controller has been designed for the 2-DOF helicopter using the Linear Quadratic Regulator (LQR) technique. The following simplifying assumptions are made in order to calculate dynamics equations of the system: (1) vibration in blades is zero, (2) the rotational speed of the helicopter is negligible compared to that of the propellers, and (3) that aerodynamic effects due to the movement of the helicopter are insignificant. The governing dynamics equations, derived using Lagrangian Mechanics, are then given by, [35]:

\[
\begin{align*}
(J_{eq,p} + m_{hel}(l_{cm})^2) \dot{\theta} \\
= K_{pp}V_{m,p} + K_{py}V_{m,y} - m_{hel}g(l_{cm} \cos \theta - B_p \dot{\theta}) - m_{hel}(l_{cm}^2 \sin \theta \cos \theta \dot{\psi}^2)
\end{align*}
\] (2.1)

\[
\begin{align*}
(J_{eq,y} + m_{hel}(l_{cm}^2 \cos^2 \theta)) \dot{\psi} \\
= K_{yp}V_{m,p} + K_{yy}V_{m,y} - B_y \dot{\psi} + 2m_{hel}(l_{cm}^2 \sin \theta \cos \theta \dot{\theta}) \dot{\psi}
\end{align*}
\] (2.2)
The equations can be linearized around the operating point or home position \( \theta = \psi = \dot{\theta} = \dot{\psi} = 0 \) as:

\[
(J_{eq,p} + m_{heli}l_{cm}^2)\dot{\theta} = K_{pp}V_{m,p} + K_{py}V_{m,y} - B_p\dot{\theta} - m_{heli}gl_{cm} \tag{2.3}
\]

\[
(J_{eq,y} + m_{heli}l_{cm}^2)\dot{\psi} = K_{yp}V_{m,p} + K_{yy}V_{m,y} - B_y\dot{\psi} + 2m_{heli}l_{cm}^2\theta \dot{\theta} \dot{\psi} \tag{2.4}
\]

where, \( m_{heli} \) is the total moving mass of the helicopter, \( l_{cm} \) is the distance from the pitch axis of the center-of-mass along the body, \( g \) is gravitational acceleration, \( \theta \) is the pitch angle, \( \psi \) is the yaw angle, \( J_{eq} \) is the total moment of inertia, \( K \) is the thrust torque constant, \( V_m \) is the motor voltage and \( B \) is the viscous rotary friction. The subscripts \( p \) and \( y \) represent the pitch and yaw, respectively. The definition of pitch and yaw angles, the sign, the thrust forces acting on pitch and yaw axes (\( F_p \) and \( F_y \)), and gravitational force (\( F_g \)) are shown in Figure 2.2.

Figure 2.2: 2-DOF model helicopter dynamics model. [35]

The state-space model of the system can be derived by defining the state vector as \( \mathbf{x} = [\theta \quad \psi \quad \dot{\theta} \quad \dot{\psi}]^T \) and substituting it in the equations (2.3) and (2.4) as follows:
\[
\dot{x} = \\
\begin{bmatrix}
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & -\frac{\eta_p}{I_{eq,p}+m_{helit}l_c^2} & 0 & 0 & 0 \\
0 & 0 & 0 & -\frac{\eta_y}{I_{eq,y}+m_{helit}l_c^2} & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3 \\
x_4 \\
x_5 \\
x_6
\end{bmatrix}
+ \\
\begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\frac{k_{pp}}{I_{eq,p}+m_{helit}l_c^2} & \frac{k_{py}}{I_{eq,y}+m_{helit}l_c^2} & 0 & 0 \\
\frac{k_{pp}}{I_{eq,p}+m_{helit}l_c^2} & \frac{k_{py}}{I_{eq,y}+m_{helit}l_c^2} & 0 & 0 \\
0 & 0 & \frac{k_{pp}}{I_{eq,p}+m_{helit}l_c^2} & \frac{k_{py}}{I_{eq,y}+m_{helit}l_c^2} \\
0 & 0 & \frac{k_{pp}}{I_{eq,p}+m_{helit}l_c^2} & \frac{k_{py}}{I_{eq,y}+m_{helit}l_c^2}
\end{bmatrix}
\begin{bmatrix}
u_p \\
u_y \\
u_x \\
u_y \\
u_x \\
u_y
\end{bmatrix}
\] 

(2.5)

\[
y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x; \quad \text{where } x = \begin{bmatrix} \dot{x} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{\psi} \end{bmatrix} \text{ and } u = \begin{bmatrix} V_{m,p} \\ V_{m,y} \end{bmatrix}
\] 

(2.6)

The output \(y = [x_1, x_2, x_3, x_4]^T\) implies that all the states are being measured. However, in the actual system only the pitch and yaw angles are measured by the encoder sensors and the time derivatives of the angles (velocity of these angles) are calculated digitally.

To improve the steady-state error of the system, an integrator, I, was also added to the controller by augmenting the state vector in (2.6) with two additional states: \(x_5 = \theta\) and \(x_6 = \psi\), defined as:

\[
x_5 = \int \theta dt \quad ; \quad x_6 = \int \psi dt
\]

(2.7)

Therefore, the state-space model is given by:

\[
\dot{x} = \\
\begin{bmatrix}
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & -\frac{\eta_p}{I_{eq,p}+m_{helit}l_c^2} & 0 & 0 & 0 \\
0 & 0 & 0 & -\frac{\eta_y}{I_{eq,y}+m_{helit}l_c^2} & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3 \\
x_4 \\
x_5 \\
x_6
\end{bmatrix}
+ \\
\begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\frac{k_{pp}}{I_{eq,p}+m_{helit}l_c^2} & \frac{k_{py}}{I_{eq,y}+m_{helit}l_c^2} & 0 & 0 \\
\frac{k_{pp}}{I_{eq,p}+m_{helit}l_c^2} & \frac{k_{py}}{I_{eq,y}+m_{helit}l_c^2} & 0 & 0 \\
0 & 0 & \frac{k_{pp}}{I_{eq,p}+m_{helit}l_c^2} & \frac{k_{py}}{I_{eq,y}+m_{helit}l_c^2} \\
0 & 0 & \frac{k_{pp}}{I_{eq,p}+m_{helit}l_c^2} & \frac{k_{py}}{I_{eq,y}+m_{helit}l_c^2}
\end{bmatrix}
\begin{bmatrix}
u_p \\
u_y \\
u_x \\
u_y \\
u_x \\
u_y
\end{bmatrix}
\] 

(2.8)
The linear full-state feedback control law can be defined as \( u = -kx \). The optimal value of the control gain, \( k \), can be derived using an LQR technique that minimizes a quadratic weighted sum of the energy expenditure and deviation from equilibrium, with weighting matrices, \( Q_{LQR} \) and \( R_{LQR} \). In addition to the LQR controller, the pitch position is also regulated by a nonlinear Feed-Forward (FF) control term that compensates for the gravitational torque \(-mg_{heli}g_{cm} \cos \theta\), shown in (2.1), and applies the voltage required for hovering of the helicopter at the desired position. The Feed-forward control is given by:

\[
u_{ff} = K_{ff} \frac{mg_{heli}g_{cm} \cos \theta_d}{k_{pp}}
\]

(2.10)

where \( \theta_d \) is the desired pitch angle and \( K_{ff} \) is the feed forward control gain. It can be seen from (2.10) that the FF control term is configuration-dependent; therefore, it has to be updated as the helicopter body moves and the desired pitch angle changes.

The combination of feed-forward and the LQR with integrators provides the control law that converges the current position \((\theta, \psi, \dot{\theta}, \dot{\psi})\) to the desired position \((\theta_d, \psi_d, 0, 0)\). The FF+LQR+I control law is then written as:

\[
u = [u_{ff}] - [k_{11} k_{12} k_{13} k_{14}] \begin{bmatrix} \theta - \theta_d \\ \psi - \psi_d \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} - [k_{15} k_{16}] \int (\theta - \theta_d) dt - [k_{25} k_{26}] \int (\psi - \psi_d) dt
\]

(2.11)

\[
y = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\
\end{bmatrix} x, \text{ where } x = \begin{bmatrix} \theta \\ \psi \\ \dot{\theta} \\ \dot{\psi} \\ \int \theta dt \\ \int \psi dt \end{bmatrix}
\]

(2.9)
where $u_{ff}$ refers to the feed-forward controller and the next two terms are LQR+I controller elements. $k_{ij}$ represents the elements of the so-called gain matrix.

For visual servoing purposes, the desired pitch and yaw angles must be provided to the controller. To calculate the feedback error, the current pose of the flyer is continuously compared to the desired pose. This error signal is then used to calculate proper input voltage for pitch/yaw motors based on Eqn.(2.11). Note that the desired pitch and yaw angles of the helicopter are calculated from the visual information provided by the onboard camera.

2.2. Image processing

The applied image processing algorithm provides the current pose and diameter of the object, $d$, relative to the flyer. The object is a moving ping-pong ball and is represented by the image coordinates of its centeroid $c = \begin{bmatrix} u \\ v \end{bmatrix}$, $u$ and $v$ bring the horizontal and vertical image coordinates, respectively. The algorithm was implemented in MATLAB/Simulink. Figure 2.3 summarizes the image processing algorithm.
The camera, attached to the helicopter, provides Red-Green-Blue (RGB) images. Since the object recognition is color-based, and in order to reduce the effect of lighting condition on the results, RGB images are converted into Hue-Saturation-Value (HSV) color space. Therefore, the segmentation of the ball will be easier by setting a suitable threshold based on the object and background colors. In this study, the selected object has an orange color on a blue background, creating the best contrast. Applying the threshold on the HSV image results in a binary image, and it will be used to determine the pixel coordinates of the object’s centroid and its diameter in the image.
The buffering filter is responsible for validating the obtained current centroid. It is possible that under some lighting conditions, the image processing algorithm does not recognize the object correctly and calculates an incorrect value as the coordinates of the object's centroid. In order to resolve this issue, the calculated values are compared to those obtained in the previous sample time. If the current centroid is moved more than a reasonable threshold, the current values are rejected and the previous centre coordinates are used; otherwise, the obtained centroid pixels are valid and will be passed through the buffering filter. Here, the threshold is 50 pixels at the frame rate of 10 fps, which corresponds to the ball’s speed of 1.25 ft/s at a distance of 25 inches. Figure 2.4 compares the filtered and desired coordinates ($c$ and $c^*$). Note that the desired location, shown in Figure 2.4 (c) is the red dot in the middle of the image with a size of 320x240 pixels; therefore, $c^* = [160 \ 120]^T$.

![Figure 2.4](image)

(a) (b) (c)

Figure 2.4: The ping-pong ball's (a) acquired image, (b) segmented location, and (c) acquired image with desired and current location superimposed.

2.3. Target Depth Estimation

For analyzing the visual information provided by the camera, the depth of the image, or distance between the object and the camera projection centre, must be determined. The depth is shown with $^cZ_b$, where $c$ stands for the camera coordinates.
frame. The relationship between imagery data and the depth of the target object is achieved based on a pinhole perspective projection model with a frontal image plane.

Figure 2.5 illustrates how the object is projected on the image plane; the orange circle represents the object and \( d_b \) is the actual object diameter. The solid line is the image plane and is assumed to be located at the focal length, \( f \), from the centre of projection, \( o \). An offline camera calibration technique (as described in [39]) was used, and the focal length was determined to be 268 pixels for an image of size 320×240 pixels. The \( g-h \) line corresponds to the projection of the ball on the image plane and is shown as \( d \). The purpose is to calculate the perpendicular distance from the centre of projection to the object, which is defined as depth.

![Figure 2.5: Top view of the object (ball) and its projection on image plane](image)

Considering two similar triangles, \( o-g-h \) and \( o-i-j \), in Figure 2.5, the relation between the depth and image parameters is given by:

\[
\frac{c_{Z_b}}{d} = \frac{d_b \cdot f}{d} \quad (2.12)
\]
This equation is valid as long as the centroid of the object is in the vicinity of the image’s centre point.

2.4. Incremental motion calculation

When the helicopter and the object are moving relative to each other, the location of the object in two sequential images is different. Thus, the change in the ball position and its relation to the required incremental motion should be determined in order to compensate for that change and make the helicopter track the ball location.

![Diagram of motion of the ball from location #1 to Location #2.](image)

In Figure 2.6, the position labeled #1 is the initial position of the ball and its centroid is assumed to be in line with the centre of the image (Figure 2.6 b), so the centroid of the ball, centre of projection and pivot point of the helicopter (points $k$, $o$ and $a$, respectively) line up. In this case, the ball is moved to the position labeled #2, which is in $k-m$ distance. For simplification, positions labeled #1 and #2 are assumed to be at the
same elevation; however, the resulting equation can be extended to the general situation. The lateral distance is denoted by $l_b$ and the movement toward the camera is shown by $\Delta^cZ_b$. These motions can be mapped to the image plane; the lateral movement of the ball results in $\Delta u_e$ horizontal pixel error:

$$\Delta u_e = u^* - u$$ (2.13)

where $u$ and $u^*$ are the current and desired horizontal pixel coordinates, respectively. Comparing these current and desired positions gives the horizontal pixel error. In order to compensate for this error, the helicopter needs to move in the yaw direction by an incremental angle, $\Delta\psi$, in order to realign the image of the ball and the centre of the image.

Considering $\Delta^cZ_b$ is much smaller than $l_b$, the $k-m$ distance can be approximated by $l_b$. The relationship between $\Delta u_e$ and $\Delta\psi$ can then be achieved by considering two similar triangles, $a-k-m$ and $o-k-m$, expressed as:

$$\frac{f}{\Delta Z_b} \approx \frac{\Delta u_e}{l_b}$$ (2.14)

Combining (2.12) and (2.14), one can write:

$$l_b \approx \frac{d_p \Delta u_e}{d}$$ (2.15)

and $k-m$ can be approximated by a small arc, so:

$$l_b \approx \Delta\psi(r + ^cZ_b)$$ (2.16)
where $r$ is the distance between $a$ and $o$ points. The following relation between
the incremental yaw angle and the object and image parameters can be achieved by
combining the (2.12), (2.15) and (2.16) equations.

$$
\Delta \psi \approx \frac{d_b \Delta u_e}{dr + d_b f} 
$$

Finally, the calculated incremental yaw angle in (2.17) is scaled by an Error
Clamp Gain (ECG) of 0.1, ensuring a stable and smooth helicopter motion [40].

$$
\Delta \psi \approx \frac{d_b \Delta u_e}{dr + d_b f} \cdot ECG 
$$

The clamped incremental angle is passed through the controller, according to
which the helicopter is adjusted. The same procedure in the pitch direction results in
equation (2.19), which indicates the relationship between the incremental pitch angle and
the vertical pixel error.

$$
\Delta \theta \approx \frac{d_b \Delta v_e}{dr + d_b f} \cdot ECG 
$$

where $\Delta \theta$ is the incremental pitch angle and $\Delta v_e$ is the vertical pixel error. The
results of [41] show that the developed vision-based control is able to successfully servo
a stationary object and track the moving target object. In the case of servoing, the error
between the current position and the desired position was converged to zero within 8
seconds, regardless of the initial position of the object and the distance between the
camera and the object. However, the results explain that fluctuation around the desired
point in the horizontal direction was more than in the vertical direction. In other words,
the controller in the yaw direction was more oscillatory than the pitch direction controller.
In the following section the possible effective parameters on the response of the system
are introduced and their effects are investigated. Thereafter, in order to improve the response of the system, the vision-based control is improved by adding a derivative control to reduce the aforementioned oscillation in the results.

2.5. Characterization and Sensitivity Analysis of Effective Parameters

Large initial error in the image can generate unwanted overshoot in the system. To avoid this, the image error was discretized and time scaled to generate a reference moving trajectory. Multiplying the calculated incremental angles by the error clamping gain (ECG) can be interpreted as planning a linear trajectory for the 2-DOF model helicopter. To parameterize this trajectory, higher order polynomials will be used in the next section to further “smoothen” the response of the flyer. In addition, different sampling rates of the image acquisition/processing were employed in several experimentations to take into account the effect of time delay in availability of the reference points. In addition to the ECG and image acquisition rate, the effect of the initial position of the object and the location of the mounted camera on the response of helicopter is also investigated.

2.5.1. Error-Clamping-Gain and Image Acquisition Rate

Optimal values of the ECG and the image acquisition sampling rate (frames per second, FPS) for different flying circumstances are obtained via experimentations, and the recorded data are analyzed offline. Based on the characteristics of the obtained results of developed the vision based control algorithm, one can consider the controller as a second order system for the sake of simplicity of the evaluation. To characterize the behavior of the system based on these parameter's values, two different sets of
experiments have been conducted. The experimental setup and the procedure of calculating the system's poles in each individual experiment are described briefly.

2.5.2. **Experimental Setup**

The 2-DOF model helicopter is equipped with a light-weighted camera for providing the images. An orange-colored ping-pong ball is attached to a blue background and it is known as the object (Figure 2.7). Two different sets of experiments have been conducted. In the first set, the ECG parameter value and the image acquisition rate are changed, and in the other set, the effect of initial position and the location of the camera are investigated. The vision-based control algorithm is meant to bring the object from its initial position in the image to the centre. The initial position of the object is set to be constant to ensure that it does not affect the response of the system. The effect of the initial position of the object and the location of the camera is investigated separately in another set of experiments. The following steps are followed to calculate the system’s poles:

1. The ECG values for pitch and yaw movement are changed within the range of [0.01, 0.1], and the sampling rate (FPS) for image acquisition is changed from 2 Hz to 30 Hz.

2. The second order response of the system is recorded for each parameter set. This information is used to calculate the system's poles in the S-domain.

3. Because of the sampled imagery information provided for the control algorithm, the whole system is assumed to be a digital system. Therefore, the calculated poles in the S-domain along with the sampling rate $T=1/$FPS are used to calculate the system’s poles in the Z-domain using $Z=e^{-ST}$.

4. The values of the poles calculated in the Z-domain are plotted against the normalized frames-per-second and the normalized error-clamping-gain. The
optimal values can be selected based on the expected/acceptable speed and overshoot of the response of the system.

In the second set of experiments, the initial location of the object and also the location of the camera have been varied. The poles of the system are found using the aforementioned procedure, and their response with respect to variation of these parameters is investigated. The results are provided in the next section.

![Experimental setup for evaluating the developed vision-based algorithm](image)

**Figure 2.7: Experimental setup for evaluating the developed vision-based algorithm**

### 2.6. Results and discussion

#### 2.6.1. Error clamping gain (ECG) and Image acquisition rate (FPS)

Figure 2.8 (a) and (b) show the 3D plots of closed-loop poles of the system in the Z-domain versus the normalized error clamping gain and the normalized sampling rate for two sets of experiments with 25 and 36 inches distance, respectively. The similarity of these figures shows that the distance between the object and camera does not play a significant role in the robustness of the proposed algorithm and also does not have major effect on the response of the system. Furthermore, the plots indicate that the higher
sampling rate results in a more stable system, as the poles’ amplitude is getting smaller and closer to the origin of the Z-plane. Interestingly, three local minima are present, and all of them are in the neighborhood of the 5 FPS sampling rate.

Figure 2.8: Variation of the closed-loop poles versus the sampling rate and the error-clamping gain
For better understanding of the effect of ECG, Figure 2.9 is provided, which illustrates the poles’ variation in the S-domain according to different ECG values. By decreasing the ECG, the poles are moving farther from the imaginary axis. The farther the poles are, the more stable the system is, and the overshoot value will be decreased, so that for small values of ECG an over-damped response can be achieved.

Figure 2.9: Variation of closed-loop poles versus ECG, in the S-domain.

2.6.2. Initial position of the object

The initial position of the object is another possible parameter that may affect response of the system. To investigate such effects, the object is located in several random initial positions. The 2nd order response of the system is recorded and all of the aforementioned procedures for computing the system’s poles are followed. The achieved closed-loop poles in the S-domain (as shown in Figure 2.10) are in a confined area for all
of the outspread initial positions and do not follow any meaningful trend. Comparing Figure 2.9 with Figure 2.10, one can conclude that all of the poles are at a relatively constant position.

![Figure 2.10: Variation of closed-loop poles versus for different initial positions](image)

2.6.3. **Location of the Camera**

By considering the model helicopter structure, three possible locations for the camera are selected (as shown in Figure 2.11). Because the number of the experiments is limited, the poles cannot describe the response of the system accurately. Therefore, a radial centroid error $E_c$ is defined as:

$$E_c = \sqrt{\Delta u_e^2 + \Delta v_e^2} \quad (2.20)$$

where $\Delta u_e$ and $\Delta v_e$ are horizontal and vertical pixel error, respectively. The radial centroid error implicitly represents the trajectory that the object tracked in the image plane from the current position to the desired position. Figure 2.12 shows the radial
centroid error recorded for three different locations of the camera. The similarity of the plotted errors despite the different camera locations expresses that the location of the camera does not significantly affect the behavior of the system.

Figure 2.11: Three selected location for attaching the camera
Figure 2.12: Radial centroid error, $E_c$, versus Time for (a) camera location #1, (b) camera location #2 position, (c) camera location #3.
2.7. Proportional-Derivative Controller

The developed vision-based control algorithm can be summarized and displayed as in Figure 2.13. The results of [41] showed that such an algorithm is able to bring the centroid of the object to the centre of the image and match them appropriately; however, the recorded ball trajectories show that the object has a tendency to significantly oscillate around the desired image coordinates in the horizontal direction, and an oscillatory response is achieved from the algorithm, particularly in the yaw direction.

To resolve this issue, one of the suggested methods is to vary the ECG parameter. As it was concluded in section 2.6, the smaller value of the ECG parameter provides a more stable system; the amount of overshoot and oscillation is decreased. An overdamped response is also achievable for very small ECG values (ECG=0.01). On the other hand, the small value of ECG results in a lower speed of the system.

![Figure 2.13: Vision-based control algorithm (Proportional Controller)](image-url)
The other alternative to deal with the oscillatory response is enhancing a derivative controller from the existing proportional controller. The derivative controller helps to stabilize the system without reducing the rate of responding to the actuating error [42]. The enhanced proportional-derivative (PD) controller is shown in Figure 2.14. In order to investigate the effects of an added derivative controller, the same analytical procedure as described in section 2.5.2 was followed, except that in this set of experiments, the gain value of the derivative controller \(K_d\) is selected as a variable parameter. The ECG value (proportional gain, \(K_p\)) is considered to be constant (i.e. \(ECG=0.1\)) to guarantee that this parameter is not affecting response of the system.

![Vision-based control algorithm (Proportional-Derivative Controller)](image)

Figure 2.14: Vision-based control algorithm (Proportional-Derivative Controller)

The closed-loop PD controller poles are plotted in Figure 2.15. The pole variations of the proportional controller are also displayed for the purpose of comparison. The solid blue lines represent variation of the closed-loop poles of the PD controller system and the dashed black lines illustrate the poles’ variation of the proportional controller.
By increasing the derivative controller gain ($K_d$) the poles move farther from the imaginary axis and the system becomes more stable. The results achieved from the PD controller, compared to those of the P controller, confirm the expected effect of the derivative controller on system stabilization. The enhanced derivative controller allows a higher value for the ECG parameter with no concerns about the stability, which results in a simultaneously stable and fast-responding system.

2.8. Summary and conclusion

This chapter presented the system dynamic and the development of a vision-based control algorithm for a 2-DOF model rotorcraft capable of tracking an arbitrary moving object by rotating in pitch and yaw directions. The experimental setup was addressed and the components of the proposed control structure were described. The design of tests and
the test results were analyzed and discussed. By investigating the effective parameters on response of the system, the following were concluded:

- Distance between the object and camera does not play a significant role in the robustness (of the proposed algorithm) and the response of the system.
- The system will be more stable when selecting a higher sampling rate.
- The smaller values of ECG result in less overshoot in responses of the system. By decreasing the ECG value, the closed-loop poles get farther from the imaginary axis in the S-domain, and consequently the stability increases.
- The closed-loop poles do not follow any meaningful trend for different initial positions or for camera locations. Therefore, the initial position of the object and the camera location have an insignificant effect on the response of the system.

Although a smaller value of ECG can increase the system stability, it can result in a significant decrease in the speed of the response. To overcome this issue here, a proportional derivative (PD) controller was enhanced, which resulted in a stable system without a decrease in the speed of the response.

By investigating the effective parameters on response of the system and stabilizing the controller using an additional derivative controller, the system has been characterized. Considering the experimental environment, the permitted amount of overshoot, and expected speed of the response, one can manipulate the effective parameters (i.e., $K_p$, $K_d$ and FPS) in order to obtain a desirable response.
CHAPTER 3

SIMULATOR & TRAJECTORY PLANNING FOR 2-DOF ROTORCRAFT

This chapter introduces a simulator to reproduce the behavior of the system for comparison and validation purposes. Thereafter, a new nonlinear trajectory planning algorithm is proposed (considering the system dynamics and constrains) and evaluated for the 2-DOF model helicopter.

3.1. Simulator

Simulators are efficient and cost effective alternatives for the evaluation of under-developing real-world systems and, because of their advantages, are becoming one of the important pre-construction steps in designing new structures and systems. One of the most remarkable advantages is practical feedbacks provided by the simulator, which helps designers to investigate alternative designs (without actually physically building the systems) and determine the weaknesses and strengths of the designed system to improve efficiency before the structuring phase.

By employing a simulator, the effective parameters of the system can be investigated and recognized in advance; trying all of the potential alternatives in the design phase prevents the repetition of experiments in real world, and is therefore more cost- and time-effective. It also can provide some details that are not achievable in real-world experiments. In addition, the simulators can provide more details about the system and sub-systems, and the results can be studied more carefully and step by step. This feature makes the simulator a powerful training tool that can be used for demonstrating...
the concepts of different parts of the system/algorithms or educating beginners and preparing them to operate in the real-world system as an expert.

In order to develop a simulator for a real system, it is necessary to model every operation has been done by that system by considering all resources and restrictions. Figure 3.1 illustrates the developed vision-based control algorithm in a real-world environment. Note that the ‘Trajectory Planning Algorithm’ block consists of the ‘P/PD Controller’ and ‘Incremental Angle Calculation’ blocks, which were introduced in previous chapter. Since these two blocks determine the servoing/tracking path, they provide a linear trajectory planning algorithm; later in this chapter, a more complex non-linear trajectory planning method will be introduced and evaluated.

Figure 3.1: Real-World vision-based control algorithm

‘Joint-level Controller’ and ‘Plant (Model Helicopter)’ blocks (Figure 3.1) have already been simulated and provided as the Simulink blocks by the manufacturer of the 2-DOF model helicopter (Quanser Inc.) [35]. The Simulink ‘Trajectory Planning Algorithm’ and ‘Image Processing’ blocks are developed in this study; however, some modification and adjustments are still necessary in order to make them compatible with the other blocks in
the simulator. The most significant part of developing the vision-based control simulator is providing virtual imagery information in the absence of the real camera. In the following, the process of simulating the attached camera and generating the visual data is described in detail.7

3.1.1. **Simulating the Attached camera**

This part of the simulator is responsible for providing virtual visual information based on the given initial position. The initial position of the object is assumed as a known parameter. The simulator needs to compute and update the current location of the object in the image plane based on the helicopter’s movements.

In order to generate the virtual imagery data from the initial position of the object, the real-world coordinates must be converted to the image plane pixel information. Therefore, in the first stage, two different coordinate systems (i.e. the real-world and camera coordinate systems) are defined. The real-world coordinate frame, denoted by \( W \), is stationary and its origin is assumed to be the pivot point of the helicopter. The camera frame, denoted by \( C \), is mobile and moves along with the helicopter motions. The camera frame origin is considered to be the location of the attached camera. Figure 3.2 illustrates the real-world and camera frames as well as sign convention.
The translational and rotational relationships between these two coordinate systems can be formulated as shown in equations (3.1) and (3.2), respectively.

\[
\begin{align*}
{^cT_W} &= [X,Y,Z]^T \\
{^cR_W} &= \begin{bmatrix}
\cos \psi & -\sin \psi & 0 \\
\sin \psi \cos \theta & \cos \psi \cos \theta & \sin \theta \\
-\sin \psi \sin \theta & -\cos \psi \sin \theta & \cos \theta
\end{bmatrix}
\end{align*}
\]  

(3.1) 

(3.2)

Where \( \theta \) and \( \psi \) are the pitch and yaw angles of the helicopter, respectively. Note that \( ^W T/R_c \) denotes the desirable operation vector that converts the coordinates from the world frame (\( W \)) to the camera frame (\( C \)).

The second phase is to compute the pixel information of the object from the obtained object coordination in the camera frame. For mapping from the 3-D camera frame to the 2-D image plane coordinates, a pinhole perspective camera projection model is used. \( O = [x_c \ y_c \ z_c]^T \) is the object coordination in the camera frame and the goal is
to discover the corresponding pixel coordinates in the image plane; the intrinsic parameters of the camera helps to relate these two coordinates.

Equation (3.3) formulates the relationship between the normalized camera frame coordinates and virtual pixel coordination of the object in the image plane. Note that the $kk$ matrix is known as camera matrix, whose elements consist of the intrinsic parameters of the real camera used in the real-world system. The intrinsic parameters are achieved by calibrating the camera using the *camera calibration toolbox* of MATLAB. Table 3-1 provides the definition of the intrinsic parameters of the camera.

$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = kk \begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix}$$

where

$$\begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{x_c}{z_c} \\ \frac{y_c}{z_c} \\ z_c \end{bmatrix}$$

and

$$kk = \begin{bmatrix} f_c(1) & a_c f_c(1) & cc(1) \\ 0 & f_c(2) & cc(2) \\ 0 & 0 & 1 \end{bmatrix}$$

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_c$</td>
<td>The focal length in pixels which is stored in a 2x1 vector</td>
</tr>
<tr>
<td>$cc$</td>
<td>The principle point coordinates which is stored in a 2x1 vector</td>
</tr>
<tr>
<td>$\alpha_c$</td>
<td>The skew coefficient (the angle between x and y pixel axes) which is stores in as a scalar</td>
</tr>
</tbody>
</table>
Up to this point, by calculating the pixel coordinates of the object in the image, the camera and image processing blocks of the real-world system (Figure 3.1) were simulated. The virtual pixel coordinates of the object in the image plane is compared to the desired location. Based on the resulting error, a trajectory will be planned (using the P or PD controllers) and directed to the joint-level controller and eventually to the model of the helicopter. The linear trajectory planning algorithms were already developed as Simulink blocks and used for the real system, and the joint-level controller and model of the helicopter (plant) are also provided by the manufacturer. The interconnection of these blocks creates a simulation mechanism for evaluating the real system; however, some of the restrictions must be considered.

3.1.2. **Sampling rate compatibility**

An important factor when dealing with digital signals is sampling time. For developing a simulator, an appropriate sampling time is required, so that the behavior of the real-world system can be reproduced accurately. The selected sampling time should be short enough so that it can update the information without causing any delay in the response of the system, and long enough that can be handled by the computer system. The under-developing simulator consists of three major parts:

- image acquisition/processing,
- trajectory planning algorithm, and
- joint-level controller and plant.

The sampling times for the image acquisition/processing blocks are imposed by the sampling rates of the attached camera in the real-world system. Updating the acquired visual information (based on the selected sampling time) simulates a real camera that
captures the new images in every sampling time. Therefore, the sampling time in these blocks shall be varied in different experiments when it is required.

The other sampling time in this simulator is the sampling time of the controller and model of the helicopter blocks, which are provided with the manufacturer. The sampling time for the controller and plant blocks are 1 millisecond (ms), while for the image acquisition/processing and trajectory planning algorithm blocks, it varies between 0.033 s and 0.5 s (which is equivalent to 2 to 30 frames per second of camera sampling rate). The interconnection of these blocks with their different sampling times is not feasible.

In order to connect blocks, their different sampling times should be converted to each other. The ‘Rate Transition’ block in Simulink helps to connect these blocks without interference of sampling rates. This block transits information from a lower to higher sampling rate and vice versa. To achieve the camera sampling rate from the whole simulator sampling time (1 ms), the sampling time should be transited from 1 ms to the selected sampling time. Therefore, the rate transition block holds the data and feeds them to the next block based on the selected sampling time. Figure 3.3 shows the appropriate locations for required rate transition blocks.

![Figure 3.3: Location of ‘Rate Transition’ blocks in the](image-url)
The first rate transition block (#1) converts 1 ms sampling time to the desired sampling time. The information provided to the image acquisition block is then processed, and the incremental angles are calculated. Before directing the computed angle commands to the joint-level controller, the sampling rate will be returned to 1 ms by the second rate transition block (#2). The combination of the aforementioned blocks and the algorithm provides a simulator that is accurate and compatible with the real-world system. To examine the reliability of the simulator and ensure that the responses will be accurate and compatible (in terms of quality and quantity) with the real responses, an evaluation process is required. The following part presents the examination procedure and the attained results.

3.1.3. Evaluation of the Simulator

The final step, before launching every designed and developed system, is to examine and validate the system behavior to ensure the reliability of results and evaluate the response of the system. In this study, for validating the developed simulator, the real experiments (presented in the previous chapter) are repeated with the simulation program. Since the whole vision-based algorithm is simulated and all of the effective factors are considered in the design phase, comparison between the real and simulated systems can provide a good measure of the accuracy of the developed simulator.

The previous experiments related to investigating the effect of changing parameters in proportional and proportional-derivative controllers are re-conducted when employing the simulator. The simulated system is assumed as a second-order system (same as the real system) and the same procedures (as described in section 2.5.2) for computing the poles of the system are followed.
3.1.3.1. Proportional Controller (effect of ECG)

For this set of experiments, only the proportional control is activated and the proportional controller gain (ECG) has been changed within the range of [0.01, 0.1]. The second-order response of the system is recorded for each value of ECG, and the poles of the simulated system are found. As it was concluded in the previous chapter, for a real system, the smaller the ECG value selected, the more stable the system becomes. Figure 3.4 represents the calculated poles of the simulated proportional controller versus varying the ECG parameter in the S-domain. In order to have a comparison between the real and simulated results, the previously calculated poles of the real system are also included.

![Figure 3.4: Variation of closed-loop poles of the simulated and real system versus ECG value, in S domain](image)

The poles of the simulated system are distanced from the imaginary axis as the ECG value decreases. This means that smaller values of ECG result in a more stable
system. Comparison between the poles of simulated and real systems shows that changing the ECG value has the same effect on both the simulated and real-world systems. However, the poles of the simulated system are farther from imaginary axis, meaning that the simulated system is more stable than the real system. As it was also addressed in [35], the simulated controller and model of the helicopter are more stable than the real system; it can result in more stability in the vision-based control algorithm, which was developed in this study.

3.1.3.2. Proportional-Derivative Controller

The other set of experiments (also conducted for the real system) are to explore the effect of varying the derivative controller gain values ($K_d$). In the real system, for a constant value of proportional controller gain, by increasing the derivative controller gain, the response of the system transformed from an under-damped to an over-damped response and resulted in a higher stability. The real experiments are repeated for the simulated system. All of the parameters and conditions are varied based on the real experiments, and the radial centroid error ($E_c$) is recorded and plotted for each value.

Figure 3.5 illustrates the radial centroid error plots for every derivative gain value within the range of [0.01, 0.1]. By increasing value of the gain, the overshoot value decreases, showing that the system is becoming more stable. For $K_d > 0.03$ an over-damped response is observed. Achieved simulated results verify the compatibility of the developed simulator and the real-world vision-based control algorithm.
Figure 3.5: Radial centroid error (Ec) versus time plotted by varying the Kd parameter within [0.01, 0.1] interval while ECG=0.

It is worth mentioning that, although the responses of the systems to the varying gains are similar for the real and simulated systems, the results show that the simulated system is more stable than the real-world system. This must be taken into account when transferring the results and values from the simulated to a real system. The gain values and parameters should be selected wisely to prevent any over-reaction from the real system.

Now, the simulator is ready to be used for investigating the new algorithms. Two linear trajectory planning algorithms were already implemented in the vision-based control algorithm. In the next section, a more sophisticated non-linear algorithm will be developed and evaluated.

3.2. Trajectory planning
Trajectory planning defines a path as well as a velocity and acceleration profile along that path, which not only determines the desired position in each step, but also provides the velocity and acceleration of the device in each point of the trajectory. In the trajectory planning algorithm developed in this study, dynamic constraints in the form of maximum permissible accelerations and motor voltages were taken into account. Consideration of these constraints and defining a trajectory based on them results in a predictive controller, which ensures that the planned trajectory is permissible while all of the capabilities of the system have been used.

As mentioned previously, at a lower level, a LQR was adopted as a reactive controller, where the system simply responds to its pseudo target points on a planned reference trajectory. Unlike the predictive controller that deals with steady-state disturbances, the reactive controllers overcome unknown and random instantaneous behavior of the system.

The goal is to control the 2-DOF model helicopter in order to servo it to a target configuration defined with respect to an object using image information (AKA, vision-based target locking). The flying device must remain in its position by using the simulated visual information so that the centre of the object and the image can be matched. The step-by-step methods are described below.

- The rotation matrix and camera projection model blocks convert the object position (spatial information) in the world frame to visual information (i.e., in pixel format).

- Such visual information, compared with the desired point (centre of the image), was used to calculate the required incremental pitch and yaw angles. By adding these values to the current position of the helicopter, the final desired angles for each degree of freedom were calculated.
• A quintic polynomial trajectory is planned (for a desired travelling time) to navigate the system from the initial to the final position.

• The desired pitch/yaw angles calculation block uses the planned trajectory to calculate the position of the device for each time step, which are fed to the controller as position commands (set-points). Whenever the selected travelling time (Tf) is over, the last set-point is dwelled until the system settles. This block also produces a flag in case a trajectory re-planning is necessary, and enables the trajectory planning block to plan a new trajectory.

• The LQR controller block employs the set-points obtained in the previous step to navigate the helicopter to those desired positions one at a time. The required voltages for the DC motors are calculated in this block.

• The plant (2-DOF model helicopter) moves by applying the voltages obtained in the previous step, and this procedure is repeated until the error value (the difference between centre of the image and the centre of the object) reaches a pre-set threshold value.

Figure 3.6 shows a schematic of the system’s block diagram.

Figure 3.6: Trajectory planning algorithm block diagram.
3.2.1. **Trajectory Planning Algorithm**

A trajectory will be calculated at this stage. This trajectory can be uniquely defined by its geometric path and also by the velocity/acceleration along the path. A continuous acceleration yields a $C_2$ smooth trajectory. Depending on the application, different trajectories can be implemented with this algorithm. In this study, a trajectory is planned, and re-planned when needed, satisfying these conditions:

- dynamics constraints in form of permissible acceleration, and
- a smooth trajectory which is twice differentiable (AKA, $C_2$ continuous).

Quintic polynomials are the industry standard to generate trajectories that would satisfy the abovementioned constraints, (e.g., [43]). In short, the three main reasons for adopting Quintic polynomial-based trajectories were: (1) they are the lowest order polynomial which has $C_2$ continuity, (2) re-planning can be done quickly, and (3) they are considered as the industry standard; therefore, they are widely known and studied.

In order to define a quintic polynomial trajectory, the initial and final values for position, velocity and acceleration need to be defined, while the final time value ($T_f$) needs to be chosen considering the design factors (which will be addressed in detail toward the end of this section). $T_f$ is defined as the overall traveling time. Eqn. (3.4) shows the position, velocity, and acceleration for quintic polynomials. The six coefficients of the quintic polynomial can be calculated by using (3.5), where $(q_0, v_0, a_0)$, $(q_f, v_f, a_f)$, and $T_f$ denote the initial position/velocity/acceleration, final position/velocity/acceleration, and the travel time, respectively.

\[
q(t) = c_5t^5 + c_4t^4 + c_3t^3 + c_2t^2 + c_1t + c_0
\]

\[
v(t) = \dot{q}(t) = 5c_5t^4 + 4c_4t^3 + 3c_3t^2 + 2c_2t + c_1
\]

(3.4)
\[ a(t) = \ddot{q}(t) = 20c_5t^3 + 12c_4t^2 + 6c_3t + 2 \]

Also, \( q, v \) and \( a \) denote the position, velocity and acceleration, respectively, and \( t \) denotes the time. The \( c_i \)'s are the unknown constant coefficients of the quintic polynomial.

The above equation can be written in a matrix form as:

\[
\begin{bmatrix}
1 & t_0 & t_0^2 & t_0^3 & t_0^4 & t_0^5 \\
0 & 1 & 2t_0 & 3t_0^2 & 4t_0^3 & 5t_0^4 \\
0 & 0 & 2 & 6t_0 & 12t_0^2 & 20t_0^3 \\
1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 \\
0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & 5t_f^4 \\
0 & 0 & 2 & 6t_f & 12t_f^2 & 20t_f^3
\end{bmatrix}
\begin{bmatrix}
c_0 \\
c_1 \\
c_2 \\
c_3 \\
c_4 \\
c_5
\end{bmatrix}
= 
\begin{bmatrix}
q_0 \\
v_0 \\
a_0 \\
q_f \\
v_f \\
a_f
\end{bmatrix}
\tag{3.5}
\]

Given that two reference trajectories for pitch and yaw movements would be needed, we generate two different quintic-polynomial-based reference trajectories. The trajectory must be re-planned under two conditions:

- The controller is not able to track the pre-defined trajectory: If the controller fails to follow the planned trajectory and the error between the planned trajectory and the current state of the flying device starts to diverge, the re-planning algorithm will be activated, which would plan a new trajectory based on the current state (i.e., position, velocity and acceleration) of the model helicopter.
- The object starts moving: Tracking a moving target would require constantly updating the desired configuration of the flyer; therefore, the terminal position/velocity/acceleration information used in generating the quintic polynomial-based trajectory must be revisited.

The main purpose of re-planning is to correct the flyer’s movement when needed without compromising the optimality and smoothness of its motion towards a goal configuration.

3.2.2. **Optimal Trajectory Planning**
The shortest permissible time based on the kinematic/dynamics constraints is calculated each time planning and/or re-planning a trajectory. The following optimization problem was solved to plan a trajectory. The same procedure was adopted to re-plan a trajectory as well:

\[
\min_u \{t_f\} \tag{3.6}
\]

(1) Subjected to dynamic constraints:

\[
D(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = u \tag{3.7}
\]

where \(D\), \(C\), and \(g\), denote the inertial, centrifugal/centripetal, and gravitational matrices, respectively. Also, \(u\) denotes the multiple control input vector, which consists of the input voltages to the pitch and yaw DC motors in our 2-DOF model helicopter.

(2) Subjected to kinematic constraints:

\[
q(t) = c_5t^5 + c_4t^4 + c_3t^3 + c_2t^2 + c_1t + c_0 \tag{3.8}
\]

where \(q(t)\) denotes the pitch/yaw vector.

And boundary conditions:

\[
q(t = t_0) = (p_0, v_0, a_0) \quad q(t = t_f) = (p_f, v_f, a_f) \tag{3.9}
\]

A divide and conquer search technique was adopted to solve this optimization problem. For a real-time operation, the upper bound of the travel time, \(T_f\), was limited to one obtained offline after experimenting with the real system’s performance in a human-in-the-loop control practice. For instance, a settling time of ~15 seconds was observed in the model helicopter when subjected to the maximum step change in its desired pitch and yaw angles. This value was used as the upper bound in the optimization algorithm.

### 3.3. Trajectory planning results and discussion
Three different sets of tests have been conducted in this study. In the first set, only the planning phase was implemented and the effect of the projected completion time was investigated. The purpose of the second set of experiments was to examine the effect of re-planning. Re-planning was implemented considering the same projected completion time used in the planning stage, and the resulting errors were compared. In the last set of tests, the effect of the gains selected in the low-level controller was investigated. Changing the controller gain affects the ability of the system to track the planned trajectory.

3.3.1. Set 1: Trajectory Planning (with no re-planning)

This set of tests is to show the effect of the projected completion time, $T_f$, on the response of the system. Three different $T_f$ values have been chosen (i.e., 3, 10 and 25 sec); Figure 3.7 represents the results. The first two rows (figures a-f) represent the pitch and yaw trajectories, respectively, corresponding to different values of the $T_f$. The blue dotted line represents the planned trajectory, and the red solid line the resulting (or real) trajectory (i.e., response of the system). One should note from figures that the planned/real trajectories will asymptotically merge to a straight line. This is due to the end-of-motion “padding” adopted on the reference trajectory, meaning that the final point of the planned reference trajectory gets fed back to the controller repeatedly in case the completion time is reached before the helicopter reaches its desired configuration. This would give the system enough time to settle towards its final state. The large overshoot at the beginning of the yaw movement is due to the gyroscopic coupling between pitch and yaw. As seen in Figure 3.7, by increasing the $T_f$ the deviation between the planned trajectory and the resulted trajectory (response of the controller + plant) decreases.
Table 3-2 also summarizes the error for different values of the projected completion time, namely $T_f$. For larger values of $T_f$, the error between planned resulting trajectories is less. This is due to the fact that the control gains set for the low-level controller would yield a long settling time.

**Table 3-2: Pitch and yaw Root Mean Square Errors (RMSE) for different projected completion time, $T_f$.**

<table>
<thead>
<tr>
<th>$T_f$ (seconds)</th>
<th>Pitch-Error %</th>
<th>Yaw-Error %</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0.5197</td>
<td>1.7818</td>
</tr>
<tr>
<td>10</td>
<td>0.3002</td>
<td>2.1471</td>
</tr>
<tr>
<td>25</td>
<td>0.2445</td>
<td>1.8706</td>
</tr>
</tbody>
</table>

The next row of plots (figures g-i) depicts motor voltages; the blue line shows the front motor voltage (pitch) while the red line represents the rear motor voltage (yaw). The front motor voltage can change between -24 and 24 volts, and the range for the back motor is -15 to 15 volts. As seen from these figures, the voltage values are within their acceptable ranges, which indicate that the motors have not been saturated. This confirms that the planned trajectories have been permissible and executable indeed.

The last two rows (figures j-o) illustrate the velocity and acceleration profiles for different $T_f$s. The red solid line shows pitch-velocity/acceleration profiles and the blue dotted line shows the same for yaw. These plots verify that the velocity and acceleration profiles are continuous, thus yielding smooth tracking motions in the 2-DOF model helicopter.
Figure 3.7: System’s response to planned trajectories: (a – f) Pitch/yaw trajectories, (g – i) control inputs in form of DC motor applied voltages, (j-o) velocity and acceleration profiles.
3.3.2. **Set 2: Trajectory Planning (with re-planning)**

This set of results shows the effect of re-planning. In this regard, the Least Mean Square (LMS) error between the planned and real trajectories is calculated. By comparing this error with that in first set (i.e. trajectory planning without re-planning), one can conclude that re-planning can reduce this error.

The order of the plots in Figure 3.8 is similar to that in Figure 3.7. The re-planning may yield a more accurate tracking. However, it can also generate more ripples in motion at the time the re-planning is executed. This is due to the fact a sudden change in acceleration may be required to allow permissible accelerations in the flyer. Table 3-3 summarizes the error value in test results sets 1 and 2.

**Table 3-3: Comparison between the planning and re-planning LMS error values.**

<table>
<thead>
<tr>
<th>$T_f$ (Sec)</th>
<th>Pitch-Error</th>
<th>Yaw-Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 (Re-planning)</td>
<td>0.5034</td>
<td>3.2%</td>
</tr>
<tr>
<td>3 (Planning)</td>
<td>0.5197</td>
<td>6.3%</td>
</tr>
<tr>
<td>10 (Re-planning)</td>
<td>0.2813</td>
<td>6.3%</td>
</tr>
<tr>
<td>10 (Planning)</td>
<td>0.3002</td>
<td>3.6%</td>
</tr>
<tr>
<td>25 (Re-planning)</td>
<td>0.2359</td>
<td>3.6%</td>
</tr>
<tr>
<td>25 (Planning)</td>
<td>0.2445</td>
<td>3.6%</td>
</tr>
</tbody>
</table>

One should note that the re-planning would decrease the deviation between the planned and travelled trajectories, and consequently error value; however, it would generate bumps in the control inputs, namely voltages applied to the DC motors. Nevertheless, as depicted in Figure 3.8, these ripples will remain within the predefined limits; therefore, the motors don’t get saturated. This ensures that the planned and re-planned trajectories remain dynamically permissible throughout the system’s motion.
Spikes in the velocity and acceleration plots seen in Figure 3.8 are also due to re-planning. Once the trajectory is planned, and because of the error between the predefined and real trajectories, the re-planning algorithm attempts to compensate for this error, therefore these spikes appear at the beginning of every re-planning stage.
Re-planning, $T_f = 3$ sec  

Re-planning, $T_f = 10$ sec  

Re-planning, $T_f = 25$ sec  

Pitch Trajectory: 

Yaw Trajectory: 

Actuator Voltages: 

Velocity Profiles: 

Acceleration Profiles: 

Figure 3.8: System’s response to re-planned trajectories: (a – f) Pitch/yaw trajectories, (g – i) control inputs in form of DC motor applied voltages (j–o) velocity and acceleration profiles.
3.3.3. **Set 3: Trajectory Planning with different values for the controller gain**

The control method designed for this 2-DOF model helicopter is comprised of two different parts, namely a feed forward (FF) and an LQR. The feed forward part is to compensate for the effect of the gravitational torque on pitch motion, while the LQR controller is for regulating the pitch and yaw angles to their desired values, [35]. With the LQR, one attempts to optimize a quadratic objective function, \( J \) by calculating the optimal control gain, \( k \), in a full-state feedback control law; \( u = -kx \) as follows:

\[
\min \{ f = \int (x^TQx + u^TRu)dt \} \quad \text{subjected to the system’s dynamics; } \dot{X} = AX + Bu.
\]

The weighting matrices \( Q \) and \( R \) are the design parameters that would shape the response of the system. They should be positive definite matrices (normally selected as diagonal matrices). In this paper, we assume that \( Q = \alpha I_{nxn} \) and \( R = \beta I_{mxm} \), where \( I \) denotes the identity matrix. The numerical value of \( I \) was chosen for \( \beta \) and three different values for \( \alpha \) were selected at 50, 500, and 1000. In general, the larger the value of \( \alpha \) is, the larger control gains will be. Figure 3.9 shows the results.

The effect of the controller gain on the speed of the system’s response, and correspondingly, the need for re-planning a trajectory, were investigated. In this set of experiments, the controller sampling time was set at 1 ms and the trajectory planning sampling time was set at 100 msec. Also, the completion time for the planned trajectory, \( T_f \), was kept constant (i.e., \( T_f = 10 \text{ sec for every test} \)). In Figure 3.9, results of three tests are shown. Numerical results are also summarized in Table 3-4.
Figure 3.9: The effect of the control gains on the overall performance of the system with no re-planning, (a-f) pitch and yaw responses (g-i) motor voltages, (j-o) velocity/acceleration profile.
In Figure 3.9, a through f represent the pitch and yaw trajectories corresponding to $\alpha = 50$, 500 and 1000, respectively. By increasing $\alpha$, the system tracks the planned trajectory faster and with higher precision. Table 3-4 summarizes the discrepancy between the real and planned trajectories for 6 different values of $\alpha$ based on a Least Mean Square (LMS) error criterion. Test results verify that increasing $\alpha$, and correspondingly control gains, will cause a better tracking of the planned trajectory.

Table 3-4: Least Mean Square error between the planned and real trajectory for different values of $\alpha$.

<table>
<thead>
<tr>
<th>Design factor, $\alpha$</th>
<th>Pitch-Error</th>
<th>Yaw-Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>0.7403</td>
<td>2.9230</td>
</tr>
<tr>
<td>100</td>
<td>0.4389</td>
<td>2.8165</td>
</tr>
<tr>
<td>200</td>
<td>0.3953</td>
<td>2.6496</td>
</tr>
<tr>
<td>500</td>
<td>0.3268</td>
<td>2.3374</td>
</tr>
<tr>
<td>700</td>
<td>0.3015</td>
<td>2.2053</td>
</tr>
<tr>
<td>1000</td>
<td>0.2757</td>
<td>2.0613</td>
</tr>
</tbody>
</table>

3.4. Summary and Conclusions

In this chapter a simulator was developed reproducing the behavior of the real system of the 2-DOF model helicopter. By duplicating the experimental setups and repeating the conducted tests for the real world with the developed simulator, the behavior of the simulator was examined. Proportional and proportional-derivative vision-based controllers’ results (obtained in the previous chapter) are used to evaluate the response of the simulated control algorithm by comparison between them. It shows that the reaction of both systems to varying the P and PD controller gains are similar. By reducing the proportional gain value and enhancing the derivative gain value, the system revolves to a more stable system.
In addition, a quintic polynomial trajectory planning algorithm was introduced. A class of $C^2$-continuous quintic-polynomial-based trajectories was planned at a higher level, first taking the maximum permissible acceleration of the flyer into account. At a lower level, a Linear Quadratic regulator (LQR) was used to track the planned trajectory. The re-planning was carried out under two conditions, namely (1) when the flyer fails in tracking the planned trajectory closely, or (2) the target object to track starts moving. Test results were provided on a 2-DOF model helicopter for three categories: (1) prediction, planning and execution, (2) adaptive prediction, planning and execution, which incorporates re-planning in the algorithm when needed, and (3) gain adjustment when executing a planned and/or re-planned trajectory. Tests were carried out in simulation. The time optimality and smoothness in the executed trajectories were met. The implementation of this framework for vision-based control of a free-flying 6-DOF quadcopter is the subject of next two chapters.
PART II:

VISION-BASED CONTROL OF A

6-DOF ROTORCRAFT
CHAPTER 4

6-DOF ROTORCRAFT OVERVIEW: DYNAMIC & CONTROL ALGORITHM

The second experimental platform of this research work is a 6-DOF quadrotor, used as a case study for the purpose of vision-based control. The AR.Drone, a remotely controlled flying quadrotor helicopter built by the French company Parrot, was selected for this purpose. This chapter provides an overview of the Parrot AR.Drone, with the main focus on dynamics and control algorithms.

4.1. Introduction

In 2010, the AR.Drone was publicly presented by the company Parrot in the category of video games and home entertainment. The AR.Drone is designed as a micro quadrotor helicopter (quadcopter), and its stability can be considered its most remarkable feature. Although the AR.Drone was originally built for entrainment purposes, a broader range of applications were also considered in its design [44].

Quadcopters are known to be inherently unstable [27], hence their control will be more difficult and complicated. Therefore, for the case of the AR.Drone, various types of sensors have been used in a sophisticated manner to solve this issue and design a robust and stable platform. Despite their complexity, the embedded control algorithms allow the user to produce high level commands, which make control very easy and joyful.

In this project, ease of flying, safety and fun were considered as the main purposes of designing this platform. Since this flying device was aimed to be released to a mass market, the user interface has to be very simple and easy to work with. This means that the end-user only needs to provide the high-level commands to the controller.
The role of the controller is to convert these high level commands to low-level and basic commands, deal with sub-systems, and ensure ease of flying and playing with the device. Another important concern is the device’s safety. The algorithm has to be robust enough to overcome all disturbances which may affect the response of the system to the commands in different environments and conditions. In addition, the flying device has to be capable of fast and aggressive maneuvers as a factor of enjoyment. To reach a robust and accurate state estimation, the AR.Drone is equipped with some internal sensors such as an accelerometer, a gyroscope, sonar and a camera. Integration of the sensory information and different control algorithms can result in a good state of estimation and stabilization.

The AR.Drone has to be able to fly even in the absence of some sensory information (e.g., flying in a low-light, weakly textured visual environment or with a lack of GPS data) with the following considerations:

- Absolute position estimation is not required except for the altitude, which is important for safety purposes; [44]
- For safety reasons, the translational velocity always needs to be known in order to be able to stop the vehicle at any time to prevent drifting;
- Stability and robustness are important factors;
- No tuning, adjustment and calibration shall be needed by the end-user since the operator is almost always unfamiliar with these technical issues (control technology).

In this chapter, dynamics, algorithms and control techniques behind this system are explained. Navigation methods, video processing algorithm and embedded software will be also briefly discussed.
4.2. The Parrot AR.Drone Platform

4.2.1. Aerial Vehicle

Figure 1.3 showed the typical configuration of an AR.Drone. The AR.Drone has been designed based on a classic quad-rotor, with four motors for rotating four fixed propellers. These motors and propellers create the variable thrust generators. Each motor has a control board, which can turn the motor off in case something blocks the propeller path (for safety purposes). For instance, the AR.Drone detects whether the motors are turning or stopped; if an obstacle blocks any of propellers/engines, it will be recognized, and all of engines will be stopped immediately.

The four thrust generators are attached to the ends of a carbon fiber crossing and a plastic fiber reinforced the central cross. A basket is mounted on the central part of the cross carrying the on-board electronics parts and battery. The basket lies on foam in order to filter the motor vibrations. The fully charged battery (12.5 Volts/100%) allows for 10-15 minutes of continuous flight [44]. During the flight, the drone detects the battery level and converts it to battery life percentage. When the voltage drops to a low charge level (9 Volts/0%), the drone transfers a warning message to the user, then automatically lands [44]. Two different hulls are designed for indoor and outdoor applications; the indoor hull (Figure 4.1a) covers the body to prevent scratching the walls while the outdoor hull (Figure 4.1b) only covers the basket (battery shield) to minimize the wind drag in an outdoor environment.
Each of the four rotors produces a torque and thrust (about its center of rotation), and also a drag force opposite to the vehicle's flight direction. Front and rear rotors are placed as a pair which rotate counter-clockwise while the right and left rotors rotate clockwise [44] (Figure 4.2). The quadrotor hovers (adjusts its altitude) by applying an equal rotation speed to all four rotors. Maneuvers can be achieved by changing the pitch, roll and yaw angles. Changing the roll angle results in lateral motion, and can be obtained by changing speeds of the left and right rotors in the opposite way (Figure 4.2a). Similarly, by changing the front and rear rotors’ speeds, pitch movement can be achieved (Figure 4.2b). Yaw movement is introduced by applying more speed to rotors rotating in one direction, which make the drone turn left and right (Figure 4.2c).
4.2.2. **On-Board Electronics**

The on-board electronics consist of two parts located in the basket: the Motherboard and the Navigation board. The processor, a Wi-Fi chip, a downward camera and a connector to the front camera are all embedded in the mother board. The processor runs a Linux-based real time operating system and the required calculation programs, and also acquires data flow from cameras. The operating system handles Wi-Fi communications, video data sampling and compression, image processing, sensor acquisition, state estimation and closed loop control.

The drone is equipped with two cameras: the front camera, which has a 93-degree wide angle diagonal lens and whose output is a VGA resolution (640×480) color image at rate of 15 frames per second, and the vertical camera with a 64-degree diagonal lens at a rate of 60 frames per second. Signals from the vertical camera are used for measuring the vehicle speed required for navigation algorithms. The navigation board uses a micro-controller that is in charge of making interfaces between sensors. The sensors include: 3-axis accelerometers, 2-axis gyroscope, 1-axis vertical gyroscope and two ultrasonic sensors.

Figure 4.2: Schematic of quadrotor maneuvering in (a) pitch direction, (b) roll direction, and (c) yaw direction.
Ultrasonic sensors are used for estimating the altitude, vertical displacement and also the depth of the scene observed by the downward-looking camera. The accelerometers and gyroscopes are embedded in a low-cost inertial measurement unit (IMU). The 1-axis vertical gyroscope is more accurate than the other gyroscope and runs an auto-zero function in order to minimize heading drift.

4.3. AR.Drone Start-up

This section focuses on how the AR.Drone can be launched. After switching on the AR.Drone, an ad-hoc Wi-Fi appears, so an external computer (or any other client device supporting the Wi-Fi ad-hoc [44]) can connect to it using a fetched IP address from the drone DHCP server. Thereafter, the computer will communicate with the drone using the interface provided by the manufacturer. Three different channels with three UDP ports are provided each with specific roles. [27]

- The **command** channel is used for controlling the drone; i.e., the user can send the commands of takeoff, land, calibrate the sensors, change configuration of controllers, etc. The commands are received at 30 Hz in this channel. [27]

- The **navdata** channel provides the drone with the status and preprocessed sensory data. For instance, it determines the current type of altitude controller, the active algorithm and whether the drone is flying and the sensors are being calibrated. It also provides the current pitch, roll and yaw angle, altitude, battery state and 3D speed estimates (i.e., sensory data). All of the information is updated at a 30 Hz rate. [27]

- The **stream** channel provides the video stream of the frontal and vertical cameras. In order to increase the data transfer speed, the images from the frontal camera are compressed, so the external computer receives a 320×240 pixel image. The user can switch between frontal and vertical cameras, but
images cannot be obtained from both at the same time. Switching between cameras takes approximately 300 ms, and during this transition time, the provided images are not valid. [27]

4.4. Vision and Inertial Navigation Algorithms

4.4.1. Vision Algorithm

As mentioned, the AR.Drone is equipped with two on-board cameras, the vertical and frontal cameras. Visual information, obtained from vertical camera, is used for estimating the vehicle velocity. In order to calculate the speed from the imagery data of the vertical camera, two complementary algorithms are developed and each of them can be applied in different conditions (depending on the scene content and the expected quality of their results).

The first algorithm, which is a multi-resolution method, computes the optical flow over the whole picture range and uses a kernel (e.g., by Lucas and Kanade [46]) to smooth the spatial and temporal derivatives. During optical flow computation and in the first refinement steps, the attitude change between two successive images is ignored. The second algorithm is a corner tracker by Trajkovic and Hedley [47] which finds and tracks the corners in the scene. This algorithm considers some points of interest as trackers and tracks them in next captured images by the vertical camera. The displacement of the camera and the flying device can be achieved by calculating the displacement of these trackers; however, an iteratively weighted least-squares minimization procedure is also used in this regard.

Basically, in this algorithm, a specific number of corners are detected and placed over the corner positions; in next images the new positions of the trackers will be
searched and wrongly found trackers will be ignored. The displacement of trackers can be interpreted as displacement of the AR.Drone [45].

Generally, the first algorithm will be used as a default in the scene with low contrast, and it works for both low and high speeds; however, this algorithm is less robust compared to second algorithm. When more accurate results are needed, the speed is low and the scene is suitable for corner detection, it switches to second scheme. Therefore, accuracy and a speed threshold can be the criteria for switching between the two algorithms.

4.4.2. **Inertial Sensor Calibration**

Two different calibration procedures have been implemented for this flying device: the factory calibration and onboard calibration. The low-cost inertial sensors have been used in designing the AR.Drone, which means that misalignment angle, bias and scale factors are inevitable. The effect of these parameters cannot be neglected, and more importantly, they are different in each of the sensors. Therefore, a basic factory calibration is required. Factory calibration uses a misalignment matrix between the frame of the camera on the AR.Drone and the frame of the sensor-board as well as a non-orthogonality error parameter [45].

Misalignment between the camera and the sensor-board frames cannot be completely resolved in the factory calibration stage, so an onboard calibration is also required. The onboard calibration will be done automatically after each landing to resolve any further misalignment that can occur during taking off, flying and landing. In this procedure, the goal is to keep the camera direction horizontally unchanged by finding the micro rotations in pitch and roll directions [45]. These rotation angles will affect the
vertical references as well. All of these micro-rotations are found and implemented in appropriate rotation matrices in order to keep the calibration valid [45].

4.4.3. Attitude Estimation

Inertial sensors are commonly used for estimating the attitude and velocity in the closed-loop stabilizing control algorithm. Inertial navigation performs are based on following principles and facts.

- Accelerometers and gyroscopes can be used as the inputs for the motion dynamics. By integrating their data, one can estimate the velocity and attitude angles.

- Velocity, Euler angles and angular rates relations are given by [45]:

\[
\dot{V} = -\Omega \times V + F
\]

\[
\dot{Q} = G(\Omega, Q)
\]

(4.1)

where \( V \) is the velocity vector of the centre of gravity of the IMU in the body frame, \( Q \) represents the Euler angles (i.e., roll, pitch and yaw), \( \Omega \) is the angular rate of turn in the body frame and \( F \) represents the external forces.

- The accelerometer only measures its own acceleration (minus the gravity) and not the body acceleration. The output is expressed in the body frame, so the data has to be transformed from the body frame to the inertial frame. The accelerometer's measurements are biased, misaligned and noisy, so these characteristics need also to be considered.

- The gyroscopes are associated with some noises and biases.

Note that attitude estimation algorithms do not deal with the accelerometer bias. Accelerometer bias is estimated and compensated by the vision system, as will be discussed in section 4.5., where the aerodynamics model of the drone and visual information are both used to calculate and compensate for the bias.
4.4.4. **Inertial sensors usage for video processing**

The inertial sensors have been employed to handle micro-rotations in the images obtained by the camera. The sensors' data are used to determine the optical flow in the vision algorithm. As an example, let's consider two successive frames at the frequency of 60 Hz [45]. The purpose is to find the 3D linear velocity of the AR.Drone by computing the pixels’ (trackers’) displacement. Since the trackers’ displacements are related to the linear velocity on the horizontal plane, the problem can be altered to the computation of the linear velocity (once the vertical and angular velocities are compensated for) using the data obtained from the attitude estimation algorithm. Also, a linear data fusion algorithm is used for combining the sonar and accelerometer data in order to calculate the accurate vertical velocity and position of the UAV above the obstacle [45].

4.5. **Aerodynamics Model For Velocity Estimation**

In both hovering and flying modes, estimating the accurate velocity is very important for safety reasons. For instance, the AR.Drone has to be able to go into hovering mode when no navigation signal is received from the user (hovering mode) or estimate the current velocity and correspond it to the velocity command coming from the user’s handheld device (flying mode). The vision based velocity estimation algorithm, described earlier, only works efficiently when the ground texture is sufficiently contrasted; however, the results are still noisy and are updated slower (compared to AR.Drone dynamics). A data fusion procedure is done to combine these two sources of information. When both of the sources (vision based and aerodynamics model) are available, the accelerometer bias is estimated and the vision velocity is filtered. Once the vision velocity is not available, only the aerodynamics model will be used with the last
calculated value of the accelerometer bias. Figure 4.3 illustrates how data fusion can help to achieve an accurate velocity estimation.

![Graph showing velocity estimation](image)

Figure 4.3: An example of velocity estimation: vision-based (blue), aerodynamics model (green), and fused (red) velocity estimates. [45]

The steps reaching an accurate velocity estimation are summarized here. At first, the inertial sensors are calibrated, and then the data will be used in a complementary filter for attitude estimation and calculating the gyroscope’s bias value. The de-biased gyroscope’s measurements are then used for vision velocity information and are combined with the data acquired from the velocity and attitude estimation from the vertical dynamics observer. Thereafter, the velocity estimated by the vision based algorithm is used to de-bias the accelerometer, and the calculated bias value is used to increase the accuracy of the attitude estimation method. At the end, the body velocity is obtained from the combination of the de-biased accelerometer and the aerodynamics model.
4.6. Control Structures

The control architecture and the data fusion procedure implemented in the AR.Drone platform include several nested data fusion and navigation loops. Since AR.Drone was originally designed to be in the video gaming category, the end user has to be embedded in these loops as well. The end user (pilot) has a handheld device that is remotely connected to the AR.Drone via a Wi-Fi connection and is able to send the high level commands for navigating the aircraft and receive the video stream from the onboard cameras of the AR.Drone. Figure 4.4 shows a typical view of what the user sees on the screen of his/her handheld device (ipad, iphone or other Android devices).

![Figure 4.4: A snapshot of the AR.Drone's graphical user interface.](image)

A finite state machine is responsible for switching between different modes (take off, landing, forward flight, hovering) once it receives the user's order. As illustrated in Figure 4.5. The touch screen determines the velocity set points in two directions and also the yaw rate, and double clicking on the screen is equivalent to the landing command.
When the pilot does not touch the screen, the AR.Drone will switch to hovering mode, the altitude will be kept constant and attitude and velocity are stabilized to zero.

Figure 4.6 presents the data fusion and control architecture of the AR.Drone [45]. As the figure shows, there are two nested loops in order to control the AR.Drone, the Attitude control loop and the Angular control loop. In the attitude control loop, the estimated attitude and the attitude set-point are compared, and an angular rate set-point is produced. In flying mode, the attitude set-point is determined by the user, and in hovering mode, it is set to be zero. The computed angular rate set-point is tracked by a proportional integral (PI) controller while the angular rate control loop is only a simple proportional controller.

When no command is received from the user, the algorithm will switch to hovering mode, and switching between the flying and hovering mode is obtained through the Gotofix motion planning technique. While the AR.Drone is flying, the attitude set-point is determined by the user, and for hovering, the set-point is zero (i.e. zero attitude and zero speed). The current attitude and speed in the flying mode are considered the initial points, and a trajectory will be planned in order to reach zero speed and zero attitude (hovering mode) in a short time without any overshoot. The planning algorithm is tuned so that the performances provided in Table 4-1 can be achieved.
Figure 4.5: State machine description. [45]

Figure 4.6: Data fusion and control architecture. [45]
Table 4-1: indoor and outdoor stop times for different initial speed. [45]

<table>
<thead>
<tr>
<th>Initial speed</th>
<th>Outdoor hull</th>
<th>Indoor hull</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_0 &lt; 3$ m/s</td>
<td>0.7 s</td>
<td>1.5 s</td>
</tr>
<tr>
<td>$3 &lt; U_0 &lt; 6$ m/s</td>
<td>1.0 s</td>
<td>2.2 s</td>
</tr>
<tr>
<td>$U_0 &gt; 6$ m/s</td>
<td>1.5 s</td>
<td>2.4 s</td>
</tr>
</tbody>
</table>

4.7. Summary and Conclusion

This chapter introduced and overviewed the AR.Drone as a stable aerial platform, and all of the embedded hardware and software. The hardware structure details were described and the drone dynamics model (and how different maneuvers can be achieved) was presented. The electronic equipment mounted on the AR.Drone (e.g., processing unit, sensors, Wi-Fi chip) were also described in detail. The calibration procedure for the sensors and combining their measurements for aim of estimating the state of the quadcopter were also addressed.

The navigation and control technology implemented with the AR.Drone were discussed, and different control sub-systems and nested loops in the developed algorithm were also described. Integration of the control algorithms and fused sensory information in the AR.Drone have resulted in a robust and accurate state estimation algorithm and have provided a stable aerial platform capable of hovering and maneuvering. Finally, a connection procedure between the AR.Drone and an external device and launching steps were briefly addressed. The next chapter will present the vision-based control algorithm (for the purpose of autonomous servoing and tracking the object) proposed and developed in this study.
CHAPTER 5
VISION BASED CONTROL OF 6-DOF ROTORCRAFT

In this chapter, the vision-based control algorithm developed for the 2-DOF model helicopter will be extended and applied to the previously introduced 6-DOF quadcopter (i.e. AR.Drone). The purpose is to achieve fully autonomous control of the AR.Drone for servoing an object. An image processing algorithm is developed in order to recognize the object and bring it to the centre of the image by producing appropriate commands. In addition, the distance between the flying device and the object must be kept constant.

The user only runs the developed program; the algorithm takes the drone off, finds the object, controls all six degrees of freedom using the visual information provided by the frontal camera, and finally lands the AR.Drone after a pre-determined flight period. Trajectories followed by the AR.Drone are plotted, and achieved results are analyzed. In order to evaluate the developed algorithm, the results are compared to another reliable source of information (i.e., OptiTrack system).

5.1. Lab Space Setup

The experimental environment for flying the AR.Drone is shown in Figure 5.1, which is an indoor laboratory covered by rubber floor mats as impact cushions. As already described, the visual information from the vertical camera is used to estimate the vehicle’s velocity. A more contrasted background with sharp corners results in a more accurate estimation of vehicle’s velocity and reduces the chance of deviation caused by a
lack of appropriate visual data. Therefore, a large scale (6′ x 6′) checkerboard pattern is used to provide a suitable scene for the vertical camera and the vehicle velocity estimation algorithm.

![Figure 5.1: Schematic of the experimental environment for the 6-DOF helicopter of this study](image)

The laboratory is also equipped with a motion tracking system (OptiTrack) that consists of six cameras mounted on the walls. These cameras, along with the software program, detect the Infra Red (IR) markers and track them in real time. Four IR markers are located on the AR.Drone; the set of these markers are defined as an individual object which can be localized and tracked by the OptiTrack system. Figure 5.2 shows the location of the IR markers on the AR.Drone and the defined trackable object by the OptiTrack system.
Figure 5.2: (a) Location of IR markers on the AR.Drone, (b) The triangular shaped object defined by the markers

The OptiTrack system captures and processes 100 images per second. This amount of information is transferred to the computer through cables and a high-speed USB connection. This high speed of data acquisition and transformation feature makes the OptiTrack an accurate benchmark that can be used for evaluation of a developed algorithm. The results from the developed algorithm can be compared to OptiTrack as a well-accepted system to examine the accuracy and reliability of the algorithm.

5.2. Vision-based Control Algorithm

The developed vision-based control algorithm consists of two sub-algorithms; Image processing and Control. The algorithm aims to provide a fully autonomous flight for the AR.Drone based on the visual information provided by the frontal camera. The captured images will be used to localize the AR.Drone relative to the object. A comparison between the current and desired locations provides an error, which will be used by control algorithm to plan a trajectory for compensating this error.
processing and control algorithms developed and implemented in this study are described in detail in the following sections.

5.2.1. **Image processing algorithm**

This image processing algorithm is responsible for providing the centroid and diameter of the object in the image plane. This set of information – which is given in pixel format – will be used in the next parts of the algorithm for autonomous control purposes. The centroid, \( c \), is given in image coordinates as:

\[
c = \begin{bmatrix} u \\ v \end{bmatrix}
\]  

(5.1)

where \( u \) and \( v \) are the horizontal and vertical image coordinates, respectively.

The algorithm is developed and implemented using the C++ programming language and OpenCV (Open Source Computer Vision) libraries. OpenCV is an open-source BSD-licensed library that includes several hundreds of computer vision algorithms. The developed image processing algorithm has been summarized in Figure 5.3.

![Image processing algorithm diagram](image)

**Figure 5.3: Developed image processing algorithm**
The frontal camera on the AR.Drone provides 15 RGB images per second. The captured images are transformed from RGB to HSV color space in order to ease image segmentation and object recognition. Image segmentation, by selecting the appropriate threshold, gives a binary image in which the object is determined as white pixels. Since the field of view of AR.Drone’s frontal camera is wider than the camera attached to the 2-DOF model helicopter, and also due to farther distance between the object and camera, a larger object is selected (compared to ping-pong ball object for the 2-DOF helicopter) in this part of study.

In order to find the centroid of the object from the provided binary image, the image moments are calculated. The image moment is a specific weighted average of image pixels’ intensities, which is usually used to describe objects after segmentation [48]. Eqn. (5.2) shows the formulation for calculating different image moments, where \( I(x,y) \) is the intensity of pixel \((x,y)\). In order to find the centroid of the object – which is the white pixels in the binary image – \( M_{10}, M_{01}, M_{00} \) must be computed. Eqn. (5.3) gives the centre point coordinates in the image plane [49].

\[
M_{ij} = \sum_x \sum_y x^i y^j I(x,y) 
\]  
(5.2)

\[
c = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} M_{10}/M_{00} \\ M_{01}/M_{00} \end{bmatrix} 
\]  
(5.3)

A centroid validation mechanism (as described in Chapter 2) is implemented to ensure wrong computed values do not cause any unwanted jumps. The validated centroid coordinates are passed to the diameter calculation procedure. Figure 5.4 shows the captured image by the frontal camera and the binary image achieved after segmentation.
The black dot in the middle of the object in the binary image represents the location of the computed centroid.

![Image](image.png)

**Figure 5.4:** (a) Captured image of the object and (b) Computed binary image after segmentation, the calculated centroid point is shown as the black dot.

For calculating the diameter of the object, an $N \times N$ pixel square is selected around the given centre point. The value of $N$ can be selected based on the size of the object and distance between the object and camera (here $N=100$ is chosen). The length of the longest vertical or horizontal line that passes the centre point is considered to be the diameter of the object in the image plane. The validated centroid coordinates and diameter of the object are sent to the control algorithm. The control algorithm uses this information to produce the appropriate commands and autonomously control the drone.

### 5.2.2. Control Algorithm

The implemented control method on the AR.Drone is summarized in Figure 5.5. The pilot produces the command by tilting the handheld device or touching the screen; before directing these commands to the controllers, the ‘Angle References Computation’ block converts them to meaningful angle references for the altitude controller. Two nested attitude control and angular rate control loops use the provided reference angles
and sensory information to generate the appropriate motor voltages accordingly. The attitude controller compares the provided reference (set points) and current angles and generates the angular rate set points. Finally, the angular rate controller produces the required voltages for rotors to track the pilot's commands. This section focuses on developing a vision-based control algorithm to replace the ‘Pilot’ and ‘Angle Reference Computation’ blocks.

![Diagram of the control algorithm for manually controlled AR.Drone by the Pilot](image)

**Figure 5.5: The control algorithm for manually controlled AR.Drone by the Pilot**

The vision-based control algorithm receives the current location and diameter of the object in the image plane and is responsible for calculating the angle references based on this information (what was done by the pilot and angle references calculation blocks). The vision-based control algorithm only generates the high-level commands and does not communicate with lower layer of the control part of the drone (i.e. angular rate controller). The angular set points are produced by the vision-based controller and directed to the attitude controller. Attitude and angular rate controllers perform the same as what was described earlier and produce the motor voltages.

Since the vision-based control algorithm aims to autonomously control the AR.Drone, all six degrees of freedom (three translational and three rotational degrees of
freedom) should be taken into account. Needless to say that lateral motion can be achieved by roll angle variation, and similarly, for forward/backward motion, pitch angle should be changed. Therefore, having pitch, roll and altitude movements under control (using the provided visual information) will result in a vision-based fully autonomous flight. In this study, the yaw angle is assumed to be zero. The three developed controllers (pitch, roll and altitude controllers) will be described in detail in the following sections.

5.2.2.1. Pitch (longitudinal) motion control

The developed autonomous pitch control algorithm is shown in Figure 5.6. Pitch motion, achieved by this controller, results in the desired longitudinal movement. The diameter of the object in the image plane is given to this algorithm; the ‘Target Depth Estimation’ algorithm is responsible for calculating the current distance between the drone and the object based on the provided diameter. The detail of the target depth estimation method was described in chapter 2. Eqn. (5.4) represents the formulation for calculating the current distance, where $D'$ is the current distance, $d_b$ is diameter of the object, $d$ is the diameter of the object in image plane and $f$ is the focal length of camera – obtained by camera calibration.

Figure 5.6: Autonomous control of pitch/longitudinal motion.
The calculated distance is very sensitive to the diameter of the object given in the image plane. Every pixel of the diameter in image is equivalent to about 80 mm of calculated distance. This sensitivity will be more significant when considering the fact that the value of the given diameter of the object is highly influenced by distance, lighting condition, size of the object (in the real-world) and the resolution of the camera. In order to resolve this issue, a simple Low Pass Filter (LPF) is implemented to prevent any fluctuation in calculated distances. An N-point moving average filter (N is selected to be 7 in this study) smoothes the distance variation and avoids any unwanted jumps in the results.

The calculated distance (after averaging) is compared to the desired distance (selected to be 1500 mm). The resulted error $E_D$, given by Eqn. (5.5), will be used by a Proportional-Integral-Derivative (PID) controller to generate and direct the commands to the attitude controller of the AR.Drone. The PID controller produces the appropriate pitch angle ($\theta$) based on the given error value, $E_D$ as shown in Eqn. (5.6), where $k_p^p$, $k_i^p$ and $k_d^p$ are proportional, integral and derivative control gains, respectively. Superscript ‘$p$’ implies that these parameters are the gain values of the pitch controller.

$$D' = \frac{d_b \cdot f}{d} \quad (5.4)$$

$$E_D = D^* - D \quad (5.5)$$

$$\theta = k_p^p E_D + k_i^p \int E_D \, dt + k_d^p \dot{E}_D \quad (5.6)$$
5.2.2.2. Lateral motion control

The lateral controller controls the lateral movements by applying the proper roll angle to the drone. The centre point’s horizontal coordinate provided by image processing algorithm and the calculated distance (between camera and object), given by the target depth estimation algorithm, are used to calculate the error value in the horizontal direction, as illustrated in Figure 5.7 and formulated in Eqn. (5.7). Since the yaw angle is assumed to be zero ($\psi = 0$), the drone lateral movement must be equal to the calculated horizontal error. The calculated error value will be used by a PID controller to create the desired roll angle ($\phi$) for compensating for the lateral error. The controller uses the equation given by (5.8) to generate the required left/right movement. The roll angle calculated by the PID controller will be directed to the drone attitude controller as the set point. The lateral (roll angle) motion algorithm is shown in Figure 5.8.

\[ E_X = X^* - X = \frac{\Delta u_e \cdot D}{f} \tag{5.7} \]
\[ \phi = k_p^r E_X + k_i^r \int E_X \, dt + k_d^r \dot{E}_X \tag{5.8} \]

Figure 5.7: Diagram of the ball, current and desired positions and resulted lateral error, EX.
5.2.2.3. Vertical motion Control

Vertical and lateral controllers follow a very similar procedure. The centre point’s vertical coordinate is compared to the desired coordinate. The resulting error and the calculated distance are used to find the required displacement for the AR.Drone in the vertical direction. The interface provided by the manufacturer accepts the vertical speed \( V_y \) by the command and produces the required motor voltages accordingly. Eqns. (5.9) and (5.10) show the formulation for calculation of the vertical speed based on the given distance and vertical pixel error. Note that \( k_p^v \), \( k_i^v \) and \( k_d^v \) are PID controller gain parameters, and \( V_y \) is the required vertical speed. The developed vertical control motion algorithm is illustrated in Figure 5.9.

\[
E_y = Y^* - Y = \frac{\Delta v_e \cdot D}{f}
\]

\[
V_y = k_p^v E_X + k_i^v \int E_X \, dt + k_d^v \dot{E}_X
\]
5.2.2.4. Gain range calculation

The calculated and directed commands to the controller of the AR.Drone are allowed to vary within the range of \([-1, 1]\). Therefore, the controller gain parameters of all of the three developed controllers (longitudinal, lateral and vertical) are required to be chosen in a way that the calculated commands fit in this interval. It worth mentioning that the given range for the commands corresponds to the largest value of commands that can be handled by the drone in both indoor and outdoor environments. Since the flight area of this experiment is small, such large values of commands cannot be applied to the drone; therefore, a new range is required for this specific lab space. In this regard, the drone was flown in the lab area, controlled by the joystick, and the generated commands were recorded and analyzed. The achieved range for the commands is used to find the proper values for the developed proportional, integral and derivative controller gains.

5.3. Design of Tests

The vision-based control algorithm has been developed in the C++ programming language. OpenCV libraries and built-in commands are also applied. The interface provided by the manufacturer has been used to communicate with the drone. The
AR.Drone and the client device (a Windows-based Personal Computer, PC) are connected via a Wi-Fi connection, by which the video stream, commands and navigation data are transmitted and received.

The frontal camera of the AR.Drone provides visual information at the rate of 15 Hz, imagery information is processed and the required commands (pitch, roll and yaw angles and vertical speed) are generated. The commands are sent to the drone at the rate of 30 Hz. A higher speed of transmitting the commands guarantees that the commands are sent to the controller before a new set of visual information is received from the drone.

To examine the response of the vision-based algorithm, the drone is flown autonomously in the laboratory environment. The user runs the program only at the beginning. The developed algorithm causes the drone to take off; a few seconds are reserved for the drone before directing the commands to pass its initial fluctuation. After this period, generated commands for autonomous flight are sent to the drone. The duration of flight can also be determined in the program by the user, and a landing command will be sent to the drone after the flight time is over. In case the drone could not find the object within the predefined flight duration, the drone will land.

In the following section, the experimental results are presented. The drone is flown from different initial positions and the recorded visual information is analyzed. For evaluating the accuracy of the achieved results, they are compared to the information provided by the OptiTrack system.

### 5.4. Results and discussion

In the first set of results, the drone is flown for a pre-determined flight period in order to servo the defined object (i.e. a red ball). For the stationary object, the drone shall
hover in front of the object so that centroid of the object is matched to the centre of the captured image by the frontal camera, and the desired distance between the object and drone is set to be 1500 mm. The recorded data are shown in Figure 5.10 and Figure 5.11; horizontal, vertical and distance errors and generated commands by the PID controllers. The information is obtained based on the provided visual information.

Figure 5.10 represents the calculated command (roll and pitch angles and vertical speed) values by the developed vision-based control algorithm. Note that the range of commands in these experiments is much less than the permitted interval ([−1, 1]), it implies that how minuscule the movements will be when flying the AR.Drone in a small bounded area (e.g., the lab space in this study).

![Figure 5.10: Generated commands by the vision-based control algorithm for (a) Roll, (b) Pitch and (c) Vertical speed.](image)

Figure 5.11 illustrates the recorded error values for four trials of the servoing experiment. The initial position of the drone and distance from the object location is
different in every trial. In these experiments, the AR.Drone takes off at t=6 sec, and 3 seconds are reserved for passing the initial fluctuations; at t=9 sec the commands are generated and directed to the drone. The results show that all three of the controllers (lateral, vertical and longitudinal controllers) were successful, converging the error values to zero after approximately 15 seconds, when the transient response is passed. One may note that the (normalized) horizontal error values are larger than the other two errors. It shows that the lateral (roll) controller is not as fast as the other controllers, therefore the response is more oscillatory and the convergence speed is low compared to vertical and longitudinal controllers.
Figure 5.11: Recorded error values - achieved from visual information- versus time; (a) Horizontal error value (Ex), (b) Vertical error value (Ey), (c) Distance error value (Ed).
The vision-based data are also compared to the information obtained by the OptiTrack system. As explained earlier, four IR markers are located on the AR.Drone and defined as the trackable object for the OptiTrack system. Figure 5.12 compares the normalized error values obtained from visual feedback and the OptiTrack system.

In order to analyze the data, the shifting in the time axis between two sets of data in the plots below should be considered. Since the data are recorded by two different programs and they have been executed with a time delay, the plots do not have the exact same time origin. A comparison between vision-based and OptiTrack information demonstrates that the calculated data from visual feedbacks are a little more jittery. This can be due to emergent noises in the provided imagery information. As discussed earlier, image processing results can be influenced by lighting condition, resolution of the camera and distance. Any impairment in the object recognition procedure or centre point/diameter calculation will affect the below results and make them noisy.

Despite the aforementioned issues between the plotted data, these two sets of results are compatible and have a good agreement in terms of their trend and error values. Compatibility between the two sources’ results is smaller for horizontal error values compared to the other axes. It shows that the horizontal error information calculated from visual feedback has been affected and impaired. This can be caused by an unwanted yaw angle of the drone during the flight. The algorithm calculations are conducted based on a zero value for the yaw angle. Any value (even small ones) for the yaw orientation of the drone increases the error; such effects will be more significant in horizontal calculations and lateral movements. This also can explain the oscillatory and slow response of lateral controller, as discussed above. One solution for this issue is to choose an specifically-
shaped object rather than a spherical ball; by moving the flying device, the projected image of the object will change and the unwanted yaw movement can be detected.
Figure 5.12: Comparison between normalized error values obtained from vision-based algorithm and OptiTrack system; (a) Normalized horizontal error, (b) Normalized vertical error, (c) Normalized distance error.
5.5. Summary and conclusion

This chapter presented a vision-based control algorithm developed for a 6-DOF quadrotor (AR.Drone) to enable the drone with autonomous flight for servoing purposes. In this regard, an image processing algorithm and an optimized PID controller were developed and implemented. The results showed that the control algorithm is successfully capable of handling the goals of this study, which are hovering in front of the object and servoing it in a confined lab area. In order to evaluate the developed vision-based algorithm, the OptiTrack system is selected as a reliable source of information to compare with the information obtained from visual feedback. These comparisons showed good compatibility, although a small discrepancy was observed, which can be due to unwanted drift in the yaw orientation. The developed vision based control can be extended for vision-based control of other similar 6-DOF rotorcrafts.
CHAPTER 6
CONCLUSION AND FUTURE WORKS

Two case studies of optimum vision-based control were presented for a 2-DOF model helicopter and a 6-DOF quadrotor (AR.Drone). The vision-based control scheme, developed for the 2-DOF model helicopter, was characterized with respect to the parameters effective on the behavior of the system. All of possible effective parameters are considered and their influences on the vision-based algorithm were investigated. This optimized value of each parameter is determined in order to allow a vision-based controlled flight to adapt to any environment and condition.

In order to improve the developed vision-based algorithm, a derivative controller was enhanced to the system. The improved proportional-derivative controller resulted in a more stable system while maintaining the speed of response. It resulted in a system that was simultaneously stable and fast-responding.

A simulator was proposed as an evaluation tool for the previously developed vision-based control structure. This simulator was able to be used to examine suggested approaches before implementing them on the real system. The compatibility of the developed simulator and the real-world system was certified by reproducing the experiments – which have already been conducted by the real system and comparing the results.

The developed simulator was employed to introduce a new polynomial trajectory planning structure for the 2-DOF model helicopter. This algorithm plans the travelling trajectory for the helicopter from the current position to the desired one. The trajectory was planned based on the dynamics of the system and considering the limitations, which
ensured that the trajectory is permissible by the flying device and does not violate any of the constraints. The planned trajectory was chosen to be a quintic polynomial to guarantee continuity of velocity and acceleration profiles.

The trajectory planning algorithm was able to identify the necessity of re-planning the trajectory. When the object moves or the controller is not able to follow the planned trajectory, it is required to be re-planned; this scheme successfully activated the re-planning algorithm when it was required.

In the second part, as an extension of the previous control structure, a vision-based control algorithm was developed for a 6-DOF quadrotor, enabling it to autonomously fly. It controlled all 6 degrees of freedom using merely the provided visual information by the on-board camera. The introduced image processing algorithm computed the required information about the object’s location and size from the provided visual information. This set of data was used by the control algorithm to generate the navigation commands. This algorithm does not require any pre-defined flight condition or flight area information.

Taking advantage of the developed algorithm, the drone was successfully able to autonomously fly, recognize the object and servo it at the desired relative location. In order to validate the achieved vision-based information, a reliable external motion tracking system was employed. This system tracked the markers mounted on the quadrotor and provided their locations during the flight time. The similarity between the vision-based and motion tracking system confirmed the accuracy of the obtained results and reliability of the developed vision-based scheme.
6.1. Future Work

This research work can be extended by applying and evaluating the trajectory planning algorithm (developed for the 2-DOF system) on the 6-DOF quadrotor. Further research work is required to extend the image processing algorithm of this study to other applications and environments. For example, a 3-D perspective model can be made based on the provided visual information, making the UAV capable of chasing objects and avoiding possible obstacles. An extended image processing algorithm can be used for real-time geometry identification and measurement calculations.
REFERENCES


Retrieved April 01, 2013.


[38] Engel, J., Sturm, J., & Cremers, D. Accurate Figure Flying with a Quadrocopter Using Onboard Visual and Inertial Sensing. IMU, 320, 240.


- 107 -


