AERIAL EYES-QUADCOPTER IMPLEMENTING PANORAMA AND 3D RECONSTRUCTION

by

AVINASH RAMESH 2012103009
M MITHUL 2012103037
SHRIYA SASANK 2012103069

A project report submitted to the

FACULTY OF INFORMATION AND COMMUNICATION ENGINEERING

in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ANNA UNIVERSITY, CHENNAI – 25

MAY 2016

BONAFIDE CERTIFICATE

Certified that this project report titled **AERIAL EYES-QUADCOPTER IMPLEMENTING PANORAMA AND 3D RECONSTRUCTION** is the *bonafide* work of **AVINASH RAMESH (2012103009)**, **M MITHUL (2012103037)** and **SHRIYA SASANK (2012103069)** who carried out the project work under my supervision, for the fulfillment of the requirements for the award of the degree of Bachelor of Engineering in Computer Science and Engineering. Certified further that to the best of my knowledge, the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or an award was conferred on an earlier occasion on these or any other candidates.

Place: Chennai
Date:

Dr. AP Shanthi
Professor
Department of Computer Science and Engineering
Anna University, Chennai – 25

COUNTERSIGNED

Head of the Department,
Department of Computer Science and Engineering,
Anna University Chennai,
Chennai – 600025

ACKNOWLEDGEMENT

We would like to sincerely thank the entire committee consisting of Dr. V. Vetriselvi, Ms. S. Renugadevi, Ms. P. Velvizhy for constantly motivating us and identifying the areas of improvement on the project. We would like to extend our immense gratitude to our project guide, Dr. AP Shanthi for her perpetual support and able guidance which was instrumental in taking the project to its successful conclusion. Finally, we would like to thank the Head of the department, Dr. D. Manjula for providing a conducive environment and amenities to facilitate our project work.

Avinash Ramesh **M Mithul** **Shriya Sasank**

ABSTRACT

Quadcopters are one of the most popular technologies of the 21st century. They have uses including aerial photogrammetry(photography and 3d positioning of objects), military surveillance, taking panorama shots, movie-making, infrastructure inspection and delivery of products. It is also an extensive area of research, an example of which would be swarm robotics. Flight stability of drones is an ongoing area of research with algorithms such as PID and Fuzzy Logic under continuous development and refinement.

Our project delves into aerial photogrammetry i.e. taking pictures of objects/topography and reconstructing 3D representations of it. Since it is extremely difficult to take pictures or videos of places inaccessible to humans, such as caverns, waterfalls, ancient ruins, caves, etc, we can use drones to capture their images.

The drone uses the Raspberry Pi 2 microcomputer to perform all on-board operations including flight control and image-processing. It carries an IMU sensor that detects the orientation of the quadcopter and feeds it to the PID controller. The PID controller uses this data as input to automatically stabilize the quadcopter. It is controlled via WiFi using an Android App with a virtual joystick interface. It is also equipped with a camera module which allows it to take pictures of objects and terrain and perform panorama stitching and 3D reconstruction.

ABSTRACT

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ARM Advanced RISC Machine

BLDC BrushLess Direct Current

BMP BitMaP

CPU Central Processing Unit

CSI Camera Serial Interface

DOF Degrees Of Freedom

ESC Electronic Speed Controller

EXIF Exchangeable Image File Format

fps frames per second

GIF Graphic Inetrchange Format

GPIO General-Purpose Input/Output

GPS Global Positioning System

GUI Graphical User Interface

I2C Inter-Integrated Circuit

IDE Interactive Desktop Environment

IMU Inertial Measurement Unit

JPEG Joint Photographic Experts Group

JSON JavaScript Object Notation

LiPo Lithium Polonium

OS Operating System

PID Proportional-Integral-Derivative

PNG Portable Network Graphics

PWM Pulse Width Modulation

RAM Random Access Memory

RC Radio Control

RISC Reduced Instruction Set Computing

RPi Raspberry Pi

SCL System Clock Line

SDA System Data Line

SD Secure Digital

SfM Structure from Motion

UAV Unmanned Aerial Vehicle

UI User Interface

VGA Video Graphics Array

WiFi Wireless Fidelity

CHAPTER 1

INTRODUCTION

1.1 GENERAL OVERVIEW

The Quadcopter UAV is one of the fastest growing technologies of the 21st century. It has a myriad of uses and is a field of intensive research and continuous development. A quadcopter, also called a quadrotor helicopter or quadrotor, is a multirotor helicopter that is lifted and propelled by four rotors[10]. Quadcopters differ from conventional helicopters which use rotors which are able to vary the pitch of their blades dynamically as they move around the rotor hub. In the early days of flight, quadcopters (then referred to as 'quadrotors') were seen as possible solutions to some of the persistent problems in vertical flight; torque-induced control issues (as well as efficiency issues originating from the tail rotor, which generates no useful lift) can be eliminated by counter-rotation and the relatively short blades are much easier to construct.

It is a highly useful technology that can be used in aerial photogrammetry, reconstructing topographical landscapes for archaeological excavations, reconstruction of buildings and taking panorama shots for movie-making etc. Drones are actively used in real-estate photography where they provide "virtual walkthroughs" to allow buyers to see the elements of the home vividly without actually entering the site. They play a major role in all categories of surveillance from border patrol,

building security and road-traffic monitoring. The advantage is that they're small, highly mobile, quiet and easy to camouflage. Another innovative and popular use for drones is in the product/food delivery sector. There are several security and reliability concerns that challenge their feasibility but new technologies are being developed to combat these issues.

Ultimately, drone technology is advancing in leaps and bounds with every day and it is inevitable that it would soon be a part of our lives, solving many of our problems.

1.2 ABOUT THE PROJECT

The project is divided into three main modules. The first module is the quadcopter itself. The quadcopter runs on the Raspberry Pi 2[11] microcomputer which hosts a Debian-based OS (Raspbian Jessie) which allows the use of OpenCV and Python. It has four motors whose speeds are controlled using Pulse Width Modulation(PWM) technique. PWM signals are used to communicate with the ESC which controls the voltage given to the BLDC motors and thus control the speed. The RPi uses pigpio library to issue PWM signals to the ESC.

The flight dynamics including stability and orientation are detected using the IMU(Inertial Measurement Unit) Sensor which consists of an accelerometer, gyroscope and magnetometer. The IMU sensor has 9 degrees of freedom since the accelerometer, gyroscope and magnetometer have 3 axes each. The accelerometer measures the acceleration and tilt while the gyroscope measures the angular velocity. The sensor data are fed to the RPi controller which calculates the current orientation and automatically computes the required orientation of the quadcopter

using a PID algorithm[9]. The RPi runs a Python server which receives JSON-encoded altitude, motor speed and trim values which are decoded and used as input by the Quadcopter program. The quadcopter is further equipped with a camera module which takes pictures and feeds them to the RPi's image-processing program which then performs Panorama stitching and 3D Reconstruction of the captured images.

The second module performs image processing on the images captured by the RPi camera. It is divided into two parts Panorama Stitching[7] and 3D Model Reconstruction[8]. The module has been implemented using OpenCV 3.1.0 with Python support. The quadcopter takes pictures using the camera module which are fed to the RPi. OpenCV is used to stitch the images together to create a panorama or a 360 degree view of the environment. Panorama stitching allows one to take individual images of a wide physical space which cannot be covered in a single shot, finds overlapping features, matches them and stitches them into one large, combined shot. This can be used in various applications such as movie-making, aerial photogrammetry and terrain localization and mapping.

The second part of the module deals with 3D reconstruction. The reconstruction algorithm also called Structure from Motion(SfM) algorithm is used to recreate 3D point clouds of buildings or topographies whose pictures are taken by the quadcopter. The point clouds are then visualized on a laptop using a browser. The advantage of using a quadcopter for this purpose is that it can fly to inaccessible locations and take shots which would otherwise be very difficult to capture. It can therefore be used for archaeological surveys, security and dynamic reconstruction of terrain.

The third module is the Android App which is used to control the quadcopter. Usually quadcopters are controlled using Radio Controllers but this project makes use of an Android app which will have other functions apart from simply controlling the flight of the drone viz. sending JSON-encoded signals using WiFi to the Python server running on the RPi. The app allows the user to manually control the altitude of the quadcopter, change trim values(which are special values used for stabilizing the quadcopter) and additionally control the speed of individual motors. The app has a virtual joystick interface for simple user-interaction and control.

1.3 ORGANIZATION OF THE THESIS

Chapter 1 deals with the general overview and an introduction to the project. Chapter 2 discusses the related work that has been accomplished in the field of drone technology and integration of image-processing with UAVs. Chapter 3 gives a detailed description of the requirements of the project. Chapter 4 describes the System Design. Chapter 5 talks about the System Development process. Chapter 6 deals with the test cases, results and evaluation. Chapter 7 marks the end of the thesis with the project's contributions and future work.

CHAPTER 2

RELATED WORK

A significant amount of research and work has gone into the quadcopter. Algorithms for controlling the stability of the quadcopter such as PID(Proportional-Integral-Derivative) and Fuzzy Logic exist. With improvements to open source image processing libraries like OpenCV and the development of RaspberryPi microcomputer, the capability and use of drones for various activities have increased.

Alex G. Kendall et al had developed an on-board object-tracking control of a quadcopter with monocular vision [3] which made use of a RPi for processing along with a camera module and sensors as input to the OpenCV library whic performed object tracking based on the color of the objects. While our project also made use of the RPi and OpenCV for image processing, it focused largely on the flight stability and image processing modules and did not perform object tracking.

Quadcopters are predominantly controlled using Radio Control joysticks. Since Android phones are ubiquitous now, a virtual joystick interface was developed to control the flight and the image-processing functions of the quadcopter as an Android app.

Since our quadcopter's functions were panorama stitching and 3D reconstruction, an interface was required to render the results. A. Zul Azfar and Hazry Desa [2] had developed a Graphical User Interface for

monitoring the flight dynamics and orientation of a remotely-operated quadcopter using LABView software.The motion of the quadcopter could be captured on the GUI using LabVIEW software. The quadcopters processing was done on-board using Arduino microcontroller and the balancing and stability were managed using an IMU sensor. This was a path-breaking paper as it demonstrated how an unmanned aerial vehicle could be controlled from a remote station. The project did not concern itself with the stability of the drone or perform any specialized functions.

Priyanga M. and Raja Ramanan [5] had developed a UAV for Video Surveillance to prevent unauthorized soil-mining operations. It used a RPi for on-board processing of live feeds from a webcam. The videos could be viewed on a webpage in real-time using WiFi. The drone used a GPS to calculate position and follow the target. The project demonstrated how a drone could be used for practical and real-time applications such as surveillance and tracking with high accuracy.

In our project, signals from the app were sent as JSON-encoded data using WiFi to the Python server running on the quadcopter. The quadcopter itself was controlled remotely via WiFi using the Android App.

Image processing on-board the quadcopter had been implemented by Shilpashree et al [4] in their paper, Implementation of Image Processing on Raspberry Pi. They had described the purpose of image-enhancing algorithms to improve the quality of the images taken by the drone due to compromised image quality from unstable flight.

Image-enhancing algorithms such as Rudin-Osher-Fatemi de-noising model (ROF) had been implemented.

Shawn McCann's thesis on 3D Reconstruction from Multiple Images [6] had tried to identify various techniques for dense and sparse reconstructions using Structure from Motion(SfM) algorithms. He had demonstrated and implemented the complete flow of 3D Reconstruction using the open source Bundler software. Our project made use of VLFeat for SfM support and OpenCV which offers libraries for feature detection and matching. In both projects, bundle adjustment was used to minimize reprojection errors and generate sparse point clouds. While [6] rendered the point clouds using third-party tools(MeshLab), our project made use of an interactive JS library called, threeJS which renderered the point cloud on a browser.

Argentim et al wrote a paper on PID, LQR and LQR-PID on a quadcopter platform [1] compared the stability of the use of a PID, LQR or a PID tuned with a LQR loop. Our project used the PID controller loop as the LQR loop. Although it was robust and produced a very low steady state error, it had a big transition delay due to six feedback gains, which caused a lag in the time taken between sensing the angle and computing the output due to the limited computing power of the RPi.

Drone technology has advanced enormously along the lines of photography, surveillance, mapping of terrains and delivery. The mobility combined with the computational power of the RPi from these projects motivated us to build one of our own.

The next chapter describes the requirements analysis of the project.

CHAPTER 3

REQUIREMENTS ANALYSIS

3.1 REQUIREMENTS

3.1.1 Quadcopter

Hardware Requirements for the Quadcopter

1) RPi 2 microcomputer
2) 9DOF IMU Sensor
3) 4 BLDC Motors
4) RPi Camera Module
5) 4 Carbon Fibre Propellers
6) Battery
7) Quadcopter Frame
8) ESC
9) Male-Female Connecting Wires
10) 4 Propeller Guards
11) Breadboard
12) Resistors
13) Portable Charger

Software Requirements for the Quadcopter

1) Raspbian OS Jessie
2) OpenCV 3.1.0 and OpenCV 3.1.0
3) Python
4) Android Studio

Functional Requirements for the Quadcopter

1) Quadcopter should be connected to the WiFi network
2) LiPo batteries should be charged
3) The SD Card should be securely connected
4) The SD Card should have enough space
5) Quadcopter should move according to the processed flight control signals

3.1.2 Image Processing

Hardware Requirements for the Image processing module

1) RPi Camera
2) RPi microcomputer (Processing unit)

Software Requirements for the Image processing module

1) Python 2.7.6
2) OpenCV 3.1.0

Functional Requirements for Image processing module

1) The RPi should be connected to the WiFi network
2) The RPi Camera should be mounted securely on the quadcopter
3) The quadcopter must be stable enough to take fairly clear pictures
4) The camera signal from the Android app should be received by the quadcopter
5) The SD Card must have enough space to store the images taken
6) The RPi must execute the Python program without system failure

3.1.3 Android App Module

Software Requirements for the Android App module

1) Android Studio IDE
2) Terminal emulator for android

Hardware Requirements for the Android App module

1) Android phone supporting minimum API 18

Functional Requirements for the Android App module

1) Create the Mobile Hotspot
2) Send motion, orientation and camera signals from the app to the RPi

3.1.4 Analysis

The quadcopter is built using the RPi. The RPi was used because of its computational power, portability and immense open-source support. The quadcopter is built with 4 motors, propellers, an IMU sensor, ESC and a RPi native camera that is interfaced with the RPi. The Android app sends JSON-encoded signals to the RPi.The RPi hosts a Python server which decodes the signals received and triggers the appropriate Python program to fly the quadcopter and take pictures. The IMU sensor detects the orientation of the quadcopter and feeds that data to the quadcopter balancing program. The program can then use the roll, pitch and yaw data to compute the required orientation of the quadcopter. The RPi Camera takes images which are processed using OpenCV image processing library on the RPi and saved on the SD Card.

The next chapter describes the System Design.

CHAPTER 4

SYSTEM DESIGN

4.1 COMPONENTS IN THE PROJECT

4.1.1 Hardware Components

1. Raspberry Pi 2 microcomputer
2. 9DOF IMU Sensor
3. 4 BLDC Motors
4. Raspberry Pi Camera Module
5. 4 Carbon Fibre Propellers
6. Battery
7. Quadcopter Frame
8. ESC
9. Male-Female Connecting Wires
10. 4 Propeller Guards
11. Breadboard
12. Resistors
13. Portable Charger

4.1.2 Software Components

1. Raspbian OS Jessie
2. OpenCV 3.1.0
3. Python 2.7.6
4. Android Studio IDE

IMU sensor on the quadcopter and are fed as input to the balancing algorithm. This data is processed by the python program and the orientation of the quadcopter is automatically corrected using the PID algorithm.

When the camera signal is triggered, the Pi Camera begins taking pictures at an interval of 1 second and saves them on the SD Card. Once the camera is disabled, the image-processing program is executed on the captured images. The final results i.e. the Panorama image and the reconstructed 3D Point Cloud are separately rendered on a web browser using a remote laptop on the ground. Thus, the quadcopter is semi-autonomous as its flight is controlled by the user but its stability is automatically corrected. It is remotely controlled using an Android app.

The next chapter discusses the System Development of the project.

CHAPTER 5

SYSTEM DEVELOPMENT

5.1 INPUT AND OUTPUT TO SYSTEM

5.1.1 Input

Input to the system consists of sensor data by the IMU, JSON-encoded orientation data from the app and images taken by the RPi camera to the quadcopter.

5.1.2 Output

The output of the system consists of automatic stabilization of the quadcopter's flight, change in orientation of the quadcoper and image processing functions respectively.

1) An upward swipe on the left joystick results in increase in altitude of the quadcopter

2) A downward swipe on the left joystick results in decrease in altitude of the quadcopter

3) An upward swipe on the right joystick results in the quadcopter pitching forward i.e. increase in the back motors

4) A downward swipe on the right joystick results in the quadcopter pitching backward i.e. increase in the front motors

5) An swipe right on the right joystick results in the quadcopter pitching right i.e. increase in the left motors

6) An swipe left on the right joystick results in the quadcopter pitching left i.e. increase in the right motors

5.2 INPUT AND OUTPUT FOR EACH MODULE

5.2.1 Raspberry Pi-controlled quadcopter

Input

The input to Raspberry Pi is the gesture packets from the internet.

1. IMU sensor-data from the accelerometer and gyroscope regrading acceleration and angular velocity.

2. JSON-encoded orientation data from the Android app.

3. JSON-encoded image-acquisition signal data from the Android App.

Output

There are 3 kinds of outputs based on the input to the quadcopter.

1. IMU sensor data is used ny the PID algorithm to calculate the required orientation and stabilize the quad automatically.

2. Orientation (x,y, z) data is sent to the quad to make it move to the left/right or increase/decrease in altitude.

3. Image acquisition signals allow the RPi camera to click pictures and store them.

Process

Code for Python Server

```
s = socket.socket()
host = '0.0.0.0'
port = 12345
s.bind((host, port))
s.listen(5)
c, addr = s.accept()
print 'Got connection from', addr
```

```
bus = smbus.SMBus(1)
 while True:
  try:
    msg=c.recv(1024)
    data = json.loads(msg)
    if 'reset' in data:
      quadcopter.set_zero_angle()
      continue
    p = int(data['P'])
    i = int(data['I'])
    d = int(data['D'])
    height = int(data['pitch'])
    quadcopter.set_height(height)
    quadcopter.set_PID(p,i,d)
    except Exception as e:
      print e
quadcopter.stop()
c.close()
```

Code for Balancing the Quad using PID

```
while self.running:
            start_time = time.time()
            old_pitch, old_roll, old_yaw = pitch, roll, yaw
            (pitch, roll, yaw) = self.imu.read_pitch_roll_yaw()
            (_, _, gx, gy, _, ax, ay, _) = self.imu.read_all()
            axis_output = {'x': 0, 'y': 0, 'z': 0}
            [axis_output['x'],i_x]=self.compute_PID_output(self.kp_x,
               self.ki_x, self.kd_x, pitch-self.offset_x, i_x,
               old_pitch)
            [axis_output['y'],i_y]=self.compute_PID_output(self.kp_y,
               self.ki_y, self.kd_x, roll-self.offset_y, i_y,
               old_roll)
            [axis_output['z'],i_z]=self.compute_PID_output(self.kp_z,
```

```
    self.ki_z, self.kd_x, yaw-self.offset_z, i_z,
    old_yaw)
self.motor_bl.setW(int(self.height+axis_output['x']/2
  +axis_output['y']/2))
self.motor_br.setW(int(self.height+axis_output['x']/2
  -axis_output['y']/2))
self.motor_fl.setW(int(self.height-axis_output['x']/2
  +axis_output['y']/2))
self.motor_fr.setW(int(self.height-axis_output['x']/2
  -axis_output['y']/2))
end_time = time.time()
print end_time-start_time
while(end_time-start_time <= 0.02):
    end_time = time.time()
    time.sleep(0.0001)
i=i+1
```

5.2.2 Image-Processing Module

This module is split into two parts:

1. Panorama stitching
2. 3D Model Reconstruction

Input - Panorama Stitching

The input consists of images taken by the RPi camera.

Output

The input images are stitched together forming a 360 degree equirectangular panorama image.

Process

```
  image_1 = imread(path_to_image_1)
  image_2 = imread(path_to_image_2)

  (keypoints1, descriptors1) = detect_and_compute(image_1)
  (keypoints2, descriptors2) = detect_and_compute(image_2)
  matches = knn(descriptors1,descriptors2)
  points1 = new Array(keypoints1[i] for (i,j) in matches)
  points2 = new Array(keypoints2[j] for (i,j) in matches)
  homography = find_homography(points1,points2)
  (size, offset) = calculate_size(image_1, image_2, homography)
  warped_image = warp_perspective(image_2,homography)
  panorama = warped_image
  panorama[offset_x:offset_x+image_1_width,offset_y:offset_y+image_1_height]
     = image_1
```

Input - 3D Reconstruction

The input consists of images taken by the RPi camera.

Output

Sparse point-cloud representing 3D model of the images taken by the quadcopter rendered on a browser.

Process

```
images = read_images(image_path)
for each image in images:
  features = detect_features(image)
  save_features(features)
matches = {}
for each image1 in images:
  for each image2 in images:
    if image1 not equal to image2:
      count = match_features(image1.features,image1.features)
      if count > threshold:
```

```
matches[image1] = image2
for each image1 in matches:
  for match in matches[image1]:
    track = add_track(image1,image2)
    create_track_graph(track)
    im1, im2 = find_images_with_largest_matches(images)
    pts3d = bootstrap_reconstruction(im1,im2)
    for each image in images:
      if image not equal to im1 and if image not equal to
         im2:
        pts2d = get_keypoints(im2)
        pts3d = incremental_reconstruction(pts3d,pts2d)
        pts3d = bundle_adjustment(pts3d)
```

5.2.3 Android App Module

Input

Touch and click (Haptic) controlled movements on the virtual joystick interface of the app.

Output

Touch movement on the circular joystick results in change in orientation of the quad. Clicking the camera button send a signal to the RPi Camera to take pictures.

Process

```
joystick_left = new Joystick()
joystick_right = new Joystick()
joystick_left.setTouchListener()
joystick_right.setTouchListener()
camera_button.setClickListener()

camera_button.click(){
```

```
  camera.shoot()
  camera.enable()
}

while(app_running)
{
  action_left = joystick_left.touch()
  action_right = joystick_right.touch()
  distance_left = joystick_left.getDistance()
  distance_right = joystick_right.getDistance()
  switch(action_left)
  {
    case 'up': increase_height(distance_left)
               break
    case 'down': decrease_height(distance_left)
               break
    default: print 'Invalid action'
  }
  switch(action_right)
  {
    case 'up': pitch_forward(distance_right)
               break
    case 'down': pitch_backward(distance_right)
                 break
    case 'left': pitch_left(distance_right)
                 break
    case 'right': pitch_right(distance_right)
                  break
    case 'upleft': pitch_upleft(distance_right)
                   break
    case 'upright': pitch_upright(distance_right)
                    break
    case 'downleft': pitch_downleft(distance_right)
                     break
    case 'downright': pitch_downright(distance_right)
```

```
                    break
    default: print 'Invalid action'
  }
}
```

5.3 MODULES

5.3.1 Raspberry Pi 2-controlled quadcopter

The objective of the project was to build a semi-autonomous quadcopter capable of self-controlled, stable flight guided via wireless communication through an Android app. A quadcopter (Figure 5.1), also called a quadrotor-helicopter or quadrotor, is a multirotor helicopter that is lifted and propelled by four rotors. Quadcopters differ from conventional helicopters as they use rotors which vary the pitch of their blades dynamically as they move around the rotor hub.

In the early days of flight, quadcopters were seen as possible solutions to some of the persistent problems in vertical flight; torque-induced control issues as well as efficiency issues originating from the tail rotor could be eliminated by counter-rotation of the relatively short blades which were also much easier to construct.

The quadcopter utilizes a Raspberry Pi microcomputer that runs a Python server and takes care of all the on-board computation. It also holds an ESC, IMU sensor and a Raspberry Pi native camera.

The Raspberry Pi interfaces with the IMU using an I2C interface that allows it to communicate with the accelerometers, gyroscopes and magnetometers independently using only 2 pins(viz. SCL and SDA pins) by using separate addresses for each. The Pi gets the

accelerometer and gyroscope angles separately for each axis and computes on them to find out the overall orientation of the quadcopter.

The Raspberry Pi can only supply digital signals that is 0 or 1 i.e. on or off. This does not allow us to directly control the motors to vary their speed. The Electronic Speed Controller (ESC) allows such digital signals to be converted to variable voltage and allow speed regulation of motors by supplying a PWM signal to them. PWM signals are achieved by sending pulses at a certain frequency while controlling the amount of time the pulse is on in a cycle. The longer the pulse is high during a cycle, the higher the voltage supplied to the motor.

The ESC also allows an external power souce such as a battery to be used for powering the motors. Our drone uses a 10V LiPo battery. LiPo batteries are commonly used for powering drones because they have high discharge rates required to power the four motors functioning at high speed and requiring order of 20A of current.

The quadcopter's flight is controlled using a PID controller. The PID algorithm is a closed-loop feedback mechanism that is used in control systems. The controller attempts to minimize the error by adjusting control inputs (P,I,D values). In the quadcopter, the PID controller will be taking data measured by the IMU sensor and comparing that against expected values to alter the speed of the motors to compensate for any differences and maintain balance. It involves three separate constant parameters viz. the proportional (P), the integral (I) and derivative (D) values. Heuristically, these values can be interpreted in terms of time: P depends on the present error, I on the accumulation of past errors and D is a prediction of future errors, based

The RPi camera currently takes pictures all at once and produces the final Panorama and 3D point cloud. This could be refined to perform the reconstruction and panorama stitching dynamically as pictures are clicked. Various options for panorama such as Polar, Cylindrical etc can be added. With regard to 3D reconstruction, the point cloud generated is a sparse 3D reconstruction of the object. This could be extended to create dense point clouds which would be more realistic.

The drone could also be equipped with heat sensors to detect people under debris in disaster-hit areas. With the integration of these new features, the drone could be used for dynamic terrain mapping, disaster relief, surveillance and tracking.

APPENDIX A

PID Controller

The quadcopter uses a PID controller where ’P’ stands for proportional, ’I’ stands for integral and ’D’ stands for differential.

The Proportional term enables the quadcopter to receive a force proportional to the angle of deviation from the central angle as a corrective force to bring it back to its normal orientation. Thus, by giving a high proportional constant, the directive force that will be calculated for a deviation in angle will be large, however setting too large a ’P’ value will cause overcompensation resulting in oscillations that will keep increasing.

The Integral term is used to provide a corrective force in cases where the Proportional term cannot compensate for the deviation and angle. In such cases, the deviation and angle is aggregated thereby allowing the quadcopter to get back to its stable state quickly even in cases where turbulence or any external forces are at play. Similar to the Proportional term, too high an integral term will cause oscillations but of a lower frequency then those caused by the Proportional term.

The Differential term is used to provide compensation proportional to that of the change in angle of the quadcopter. Thus, having a high differential term will result in a high corrective force being applied in cases where rapid changes in angle occur. This allows the differential

term to either boost or resist the effects of the Proportional and Integral terms because in cases where there are quick angular changes, the Differential term will enable the controller to resist that change and in cases where the compensation is too fast, the Differential term will correspondingly resist fast changes.

A.1 SAMPLE PID CONTROLLER CODE

```
def compute_PID_output( self, kp, ki, kd, angle, old_i,
   old_angle, limit_p=100, limit_i=100, log=False):
       p = kp * angle
       i = old_i + ki * angle
       d = kd * (angle - old_angle)
       if p > limit_p:
          p = limit_p
       if p < -limit_p:
          p = -limit_p
       if i > limit_i:
          i = limit_i
       if i < -limit_i:
          i = -limit_i
       if log:
          log.write('\t' + str(p) + '\t' + str(i) + '\t' +
             str(d) + '\t')
       return [p + i + d, i]

   def compute_rate_PID_output( self, kpr, kp, ki, kd, gyro,
      angle, old_i, old_angle, limit_p=100, limit_i=100,
      limit_pr=100, log=False):
       p = kp * angle
       i = old_i + ki * angle
       d = kd * (angle - old_angle)
       if p > limit_p:
          p = limit_p
```

```
if p < -limit_p:
    p = -limit_p
if i > limit_i:
    i = limit_i
if i < -limit_i:
    i = -limit_i
total = kpr*(p + i + gyro)
print total,p,i,kpr,angle,gyro,kp
if total > limit_pr:
    total = limit_pr
if total < -limit_pr:
    total = -limit_pr
if log:
    log.write('\t' + str(p) + '\t' + str(i) + '\t' +
       str(total) + '\t')
return [total, i]
```

REFERENCES

[1] Lucas M Argentim, Willian C Rezende, Paulo E Santos, and Renato A Aguiar, "Pid, lqr and lqr-pid on a quadcopter platform", In *Informatics, Electronics & Vision (ICIEV), 2013 International Conference on*, pp. 1–6. IEEE, 2013.

[2] A Zul Azfar and Desa Hazry, "Simple gui design for monitoring of a remotely operated quadrotor unmanned aerial vehicle (uav)", In *Signal Processing and its Applications (CSPA), 2011 IEEE 7th International Colloquium on*, pp. 23–27. IEEE, 2011.

[3] Alex G Kendall, Nishaad N Salvapantula, and Karl A Stol, "On-board object tracking control of a quadcopter with monocular vision", In *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, pp. 404–411. IEEE, 2014.

[4] Lokesha H. K.S. Shilpashree and Hadimani Shivkumar, "Implementation of image processing on raspberry pi", In *International Journal of Advanced Research in Computer and Communication Engineering, Vol. 4, Issue 5, May 2015*, pp. 1–3. IJARCCE, 2015.

[5] Priyanga M. and Raja Ramanan, "Unmanned aerial vehicle for video surveillance using raspberry pi", In *International Journal of Innovative Research In Science and, Engineering and Technology, Volume 3, Special Issue 3, March 2014*, pp. 1–6. ICIET, 2014.

[6] Shawn McCann, "3d reconstruction from multiple images", In *Project: Computational Vision & Geometry Lab, Stanford University, 2015*, pp. 3–7. Stanford University, 2015.

[7] Wikipedia, “Image stitching — wikipedia, the free encyclopedia”, 2015. [Online; https://en.wikipedia.org/w/index.php?title=Image_stitching&oldid=693727653; accessed 15-April-2016].

[8] Wikipedia, “3d reconstruction — wikipedia, the free encyclopedia”, 2016. [Online; https://en.wikipedia.org/w/index.php?title=3D_reconstruction&oldid=714995129; accessed 15-April-2016].

[9] Wikipedia, “Pid controller — wikipedia, the free encyclopedia”, 2016. [Online; https://en.wikipedia.org/w/index.php?title=PID_controller&oldid=716008085; accessed 15-April-2016].

[10] Wikipedia, “Quadcopter — wikipedia, the free encyclopedia”, 2016. [Online; https://en.wikipedia.org/w/index.php?title=Quadcopter&oldid=714919346; accessed 15-April-2016].

[11] Wikipedia, “Raspberry pi — wikipedia, the free encyclopedia”, 2016. [Online; https://en.wikipedia.org/w/index.php?title=Raspberry_Pi&oldid=717342422; accessed 15-April-2016].