



The
University
Of
Sheffield.

Automatic
Control &
Systems
Engineering.

AUTONOMOUS NAVIGATION OF A ROTOR-CRAFT UNMANNED AERIAL
VEHICLE USING MACHINE VISION

by

Olalekan Ogunmolu

August 2012

Supervisor: Dr Tony J. Dodd

**A dissertation submitted in partial fulfilment of the
requirements for the degree of MSc in Control Systems**

Copyright © 2012 Olalekan Ogunmolu

All Rights Reserved

AUTONOMOUS NAVIGATION OF A ROTOR-CRAFT UNMANNED AERIAL VEHICLE
USING MACHINE VISION

Olalekan Ogunmolu

Department of Automatic Control and Systems Engineering

Master of Science

Executive Summary

Vision-based unmanned vehicle navigation is powerful in real-time data and video capture for ISR¹ missions, eliminating sensing error in observational data as it does not rely on time integration of measurements.

Aims and Objectives

Landing a RUAV in an indoor environment based on machine vision requires accuracy in features extraction and pose estimation of the robot with respect to the target. The aim of this project was to design an experimental RUAV, and build a vision subsystem for autonomous guidance during landing approach with objectives to:

- improve existing departmental UAV for autonomous navigation purposes; navigate to a predetermined landing target, hover above landing target during which the UAV camera acquires videos and camera frames;
- apply invariant moments, extract corners from UAV camera frames and compare results with calibrated helipad image; Learn actual helipad features if matching variance is high;
- estimate UAV pose in hovering condition; and implement simulation results on physical testbed for landing.

Achievements

The existing quadrotor was flown implementing optimized parameters, navigated to the landing target area for image acquisition and features extraction. Features of noised images compared to the reference shows

¹ ISR – Intelligence, Surveillance and Reconnaissance missions

translational convergence but 0.02° rotational error. A Restricted Boltzmann Machine was developed for pattern learning with simulation results showing an average classification error of 36%.

Conclusions / Recommendations

This dissertation presents the promising results of invariant moment features extraction using the curvature scale space corner detection methods on the landing target of a UAV robot.

Due to project deadlines and time-imposed constraints, the ego motion estimates useful for controller input design for autonomous landing of the UAV could not be developed. Background for this is provided to make implementation easy in future work.

ABSTRACT

The design and part-implementation of a computer vision algorithm is presented for the autonomous landing of a rotorcraft unmanned aerial vehicle.

The landing procedure involves two stages. First the UAV navigates from an initial position to a predetermined helipad location, maintains hover, and extracts corner features using the curvature scale space corner detector algorithm later matched to a standard calibrated helipad image by a Restricted Boltzmann Machine learning algorithm. The second stage involves high level motion state estimation and pose of aerial vehicle with respect to the landing target.

Invariant features extracted from corner-marked test and reference images showed 100% translation-axis correlation and 0.02° rotation error. A Restricted Boltzmann Machine for Pattern Learning with various test images showed 36% classification error.

Motion and pose estimation of aerial vehicle with respect to the landing target based on the differential case of the classic ego motion estimation problem is proposed in a feedback control structure for the UAV in future work.

INDEX TERMS: Image Processing, UAV, Mobile Robots, Invariant Moments, Curvature Space Scale, Restricted Boltzmann Machines, Deep Belief Networks, 8-point algorithm, Ego-Motion, Monocular Tracking, and Visual Navigation.

ACKNOWLEDGMENTS

It is such an honour to work with Dr. Tony Dodd whose guidance and support, suggestions and positive, constructive feedback have broadened my engineering knowledge and made this work exciting.

I would like to thank Anthony Whelpton for his technical insight and for helping me in the span of this project work. He was always there when I needed to build mechanical designs, safety components and hardware for indoor testbeds. I am in debt of his patience, reach-out and competence.

To “The Petroleum Technology Development Fund” who helped me fulfil my dream of undertaking this MSc program – I am grateful.

Finally, I am thankful to my family for their support throughout the course of my postgraduate degree and to everyone who made a contribution to this work.

Contents

Executive Summary	4
Aims and Objectives	4
Achievements	4
Conclusions / Recommendations	5
ABSTRACT.....	6
ACKNOWLEDGMENTS.....	7
Contents	9
List of Figures.....	13
1. Introduction	15
1.1 Historical Context of the UAV and its Development	15
1.2 Applications of UAVs	16
1.3 Classification of UAVs	17
1.4 The Quad-rotor Model.....	19
1.5 Problem Statement	21
1.6 Research Aims.....	21
1.7 Research Objectives	22
1.8 Dissertation Overview	23
1.9 Project Management and Planning	23
2. Literature Review	24
2.1 Vision-Based Landing Systems	25
2.1.1 Inspection/Monitoring Systems for Landing UAVs	26
2.1.2 Vision Servoing-Based Landing Control of UAVs	29
2.2 Corner Detection Techniques.....	32
2.3 Vision-based autonomous Landing of an Aircraft – Discussion.....	33

3. Hardware and Software Requirements	35
3.1 The ArduCopter Quadrotor	35
3.2 Communication Development	37
3.2.1 Radio Set-up.....	37
3.2.2 Telemetry Setup	39
3.2.3 Sonar Mount	40
3.2.4 Ground Station Set-up and Configuration	40
3.3 Electronic Speed Controllers (ESCs) and Propeller configuration	42
3.3.1 ESC's Calibration	42
3.3.2 Props Set-up.....	42
3.4 Results of Flight Tests.....	43
3.5 Camera Set-up and Incorporation	47
4. Visual Sensing Algorithm	48
4.1 Low-level Image Pre-Processing	48
4.1.1 Thresholding	50
4.1.2 Segmentation and Corner Detection Algorithm	56
4.2 Features Detection.....	60
5. Simulations and Experimental Results.....	64
5.1 Invariant Moments Sensitivity	67
5.2 Artificial Neural Networks	70
5.2.1 Boltzmann Machines.....	71
5.2.2 Restricted Boltzmann Machines	72
6. Conclusions and Recommendations	76
6.1 Accomplishments	76
6.2 Project Milestones and Recommendations	78
6.3 Future Work	80
6.3.1 Geometric Visualization	80

6.3.2	Framework of Coordinate Systems	82
6.33	Camera Setup and Coordinate Transformations	83
6.3.4	Algorithm Formulation	84
6.3.5	Differential Ego Motion Estimation	84
6.4	Concluding Remarks.....	85
	References	86
	Appendix A – Initial Project Gantt Chart.....	93
A.1	TASKS IDENTIFICATION AND OBJECTIVES.....	94
	APPENDIX B	99
	CMUCAM4 Automatic Paparazzi Program	99
	Python Program	100
	APPENDIX C	101
	Results of Low Level Image Analysis.....	101
	APPENDIX D	104
D.1	BOLTZMANN MACHINE WORKING MECHANISMS	104

List of Figures

9

1. FIGURE 1.1 UAV EVOLUTION OVER A 30 YEAR PERIOD; CREDIT: U.S. DEPARTMENT OF DEFENCE.....	16
2. FIGURE 1.2 THE EUROHAWK HALE UAV.....	18
3. Figure 1.3 BAE System's Silver Fox MALE UAV	18
4 FIGURE 1.5 THE EUROCOPTER EC-175 ROTORCRAFT [COURTESY: EADS, FIA2012]	20
5. FIGURE 1.6 (LEFT) FLIGHT PATH FLOWCHART. (RIGHT) VISION SYSTEM FLOWCHART..	22
6. FIGURE 3.1 LEFT (A) THE ARDUICOPTER TESTBED IN A READY-TO-FLY MODE RIGHT (B) CMUCAM4 BOARD	36
7. Figure 3.2 THE ARDUICOPTER IMU "OILPAN" SHIELD SITTING ATOP THE ARDUPILOT MEGA (APM) 2560 MICROPROCESSOR CONTROLLER (RED, BELOW) [Credit: ArduCopter Google Code]	36
8. FIGURE 3.3 THE FUTABA 7-CHANNEL 2.4GHZ SYSTEM(A-L)THE TRANSMITTER;(B-R)RECEIVER	38
9. FIGURE 3.4 DIGIMESH'S XBEE SERIES 2 ROUTER AND COORDINATOR MODULES.....	39
10. FIG 3.5 THE MISSION PLANNER GUI SOFTWARE.....	41
11. FIGURE 3.6 LEFT (A) THE ARDUICOPTER IN A 'FRAME X' FORMAT; RIGHT (B) THE PUSHER AND PULLER PROPS DIRECTION	43
12. FIGURE 3.7 AIRCRAFT INSTABILITY AT INITIAL START-UP	44
13 FIGURE 3.8 ATTITUDE AND ALTITUDE BEHAVIOUR AFTER TUNING EXPERIMENT	46
14 FIGURE 4.1: DESIGN OF LANDING TARGET AND RESULTS OF DIFFERENT IMAGE ANALYSIS PROCEDURES.....	49
15 FIGURE 4.21 SPATIAL ENHANCEMENTS FOR IMAGE SEGMENTATION.	52
16. FIGURE 4.22 EFFECT OF APPLYING THE HISTOGRAM EQUALIZATION METHOD ON GRAYSCALE IMAGE OF FIG 4.1C	53
17 FIGURE 4.3 LEFT (A) THRESHOLDED IMAGE RIGHT (B) HISTOGRAM OF IMAGE (A)	55
18. FIGURE 4.4 RESULT OF APPLYING EQUATIONS (4.5) AND (4.6) TO FIG (4.3A).	59
19 FIGURE 5.1 DIFFERENT CMUCAM4 FRAMES TAKEN WHILE ADJUSTING DIFFERENT PROPERTIES OF THE OMNIVISION CAMERA.....	66
20. FIGURE 5.2 EXAMPLE SMOOTHENED IMAGE FROM THE CMUCAM4 FRAME	67
21. FIGURE 5.3 EVALUATION OF INVARIANCE BETWEEN ACTUAL HELIPAD IMAGE AND CORRUPTED IMAGES.....	69

22. FIGURE 5.4 PRINCIPAL COMPONENT ANALYSIS OF EXTRACTED CORNER PAIRS IN HELIPAD IMAGE.....	70
23. FIGURE 5.5 INTERACTION OF VISIBLE AND HIDDEN UNITS VIA LEARNED WEIGHTS IN AN RBM	72
24. FIGURE 5.7 PATTERN CLASSIFICATION VISUALIZATIONS.....	75
25. FIGURE 6.1 GEOMETRY OF CAMERA FRAME WITH RESPECT TO LANDING PLANE [CREDIT: 12]	81
26. FIG 6.2 QUADROTOR INERTIAL AND LOCAL COORDINATE SYSTEMS.....	82
27. FIGURE 6.3- (A) CAMERA IMAGE COORDINATE SYSTEM (B) FEATURE POINTS IN ACQUIRED IMAGE	83
28. FIGURE C.1	103

1. Introduction

1.1 Historical Context of the UAV and its Development

Unmanned Aerial Vehicles (or UAVs) have had a long history of use starting in 1916 when A.M. Low developed the aerial target. During the first and second world wars, more were developed in the technology advances that followed. These were mostly used to deliver attack missions in a military capacity.

The United States Defence Department defines the UAV as “powered aerial vehicles sustained in flight by aerodynamic lift over most of their flight path and guided without an on-board crew. They may be expendable or recoverable and can fly autonomously or piloted remotely” (1). This definition includes UAVs and remotely piloted vehicles (or RPVs), with RPVs having the characteristic of being remotely piloted.

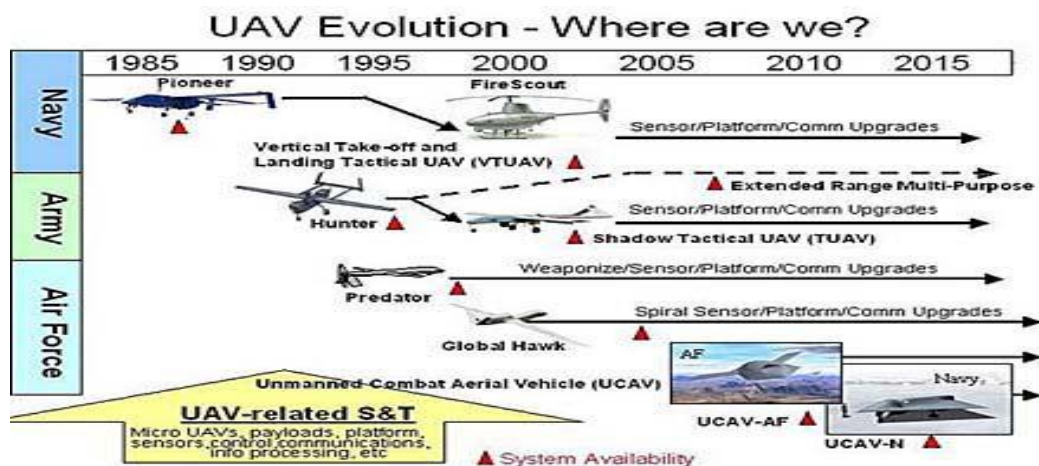
The modern class of UAVs used in military ISR² missions were first developed by Israel shortly after the end of the 1973 Yom Kippur War. In the Bekaa Valley campaign of 1982 in Lebanon ISR made available by Israeli UAVs activated Syrian air defence systems, totally neutralizing the Syrian air defences with no Israeli pilot being shot down.

This awakened wide military and political interest in UAVs in the United States as Israeli UAVs such as the Pioneer was bought and the USAF began to develop new systems. The predator UAV (RQ-1 Predator or Predator A) got developed. In 1996, the USAF deployed the system to the Balkans as an ISR tool. In 2004, the RQ-1 Predator UAV emerged as a great combat asset and

² ISR stands for intelligence, surveillance and reconnaissance missions

was used in significant military applications. By October 2007, the predator fleet had performed a “total of 300,000 flight hours, with four-fifths of this in combat missions” (1)

The MQ-1 predator was widely used in the global war on terror in “finding, fixing, tracking, targeting, engaging and assessing suspected terrorist location” [9]. Military research is presently focused on robustly integrating unmanned systems with manned and spaced systems and integrating them with other unmanned systems that include land and underwater systems.



1. FIGURE 1.1 UAV EVOLUTION OVER A 30 YEAR PERIOD; CREDIT: U.S. DEPARTMENT OF DEFENCE

But UAVs are finding applications in other areas such as private and civil use.

1.2 Applications of UAVs

UAVs find applications in broad areas ranging from counter-narcotics, agriculture and commercial applications.

In **Security** UAVs find use in public safety where they help insure officer and population safety with creative ground based aerial solutions [(2)].

Oil and Gas Assets/Pipeline installations often come into the danger of thieves, leakages or attacks. UAVs have been successfully used in the monitoring oil and gas assets [(3), (4)] lowering the cost of providing guards

over maritime oil and gas assets and helping monitor bursts/leakages in such pipeline networks.

In the **weather service**, UAVs are used in storm monitoring and tornado chasing.

Developments in micro-UAVs such as Cropcam revolutionized the way crops are managed. By using a small autopilot, camera, software and GPS module, high-resolution images are provided on demand which enables plague and disease scouting; and plant development through the aerial images provided. By advances in new composite materials and better propulsion systems, UAV sizes are reducing, enabling them to navigate environments not accessible by humans. Their long endurance provide sustained support for more efficient time-critical targeting and other missions requiring longer persistence than that provided by manned aircraft or space systems.

Micro-UAVs have the particular characteristic to get close to a target with a close-up view. Their small size, stealthy propulsion system and ability to feed data directly to battlefield airmen enhance the combat effectiveness of military missions.

1.3 Classification of UAVs

UAVS can be classified based on different characteristics among which are:

- **Wing-type**, which could be *fixed-wing* (such as commercial passenger airplanes) which use wings that produce lift as a result of the vehicle's forward aerial speed and the shape of the wings; or *rotary-wing* using rotating wings (called propellers or props) powered by electric motors to produce lift; or *orthinopters* that move in the air by the flapping of their wings (modeled after the flight behavior of birds, insects or bats);

► **Performance characteristics** where UAVs are categorized by performance factors such as engine type, power and thrust loading, endurance, speed, weight, wing-loading and range.

► **Endurance and Range:** where by determining the range and endurance in a specification/requirement, the designer is able to understand the reach of the mission objective. This also helps to determine how regularly refueling is required, how much time can be spent with the UAV carrying out its task, and how much time it needs to spend on the ground.

In this light, UAVs are generally categorized as **long, medium or short** endurance/range-based;



2. **FIGURE 1.2 THE EUROHAWK HALE UAV**



3. **Figure 1.3 BAe System's Silver Fox MALE UAV**

Tactical Unmanned Aerial Vehicles [TUAVs] are those that provide a gap filling between short-range mini-UAVs and the long-range, extended endurance MALE and HALE UAVs. They manage the flexible nature of the smaller UAVs with the longer endurance of the higher-end platforms. Typical examples include the Tracker and ATLANTE.



Figure 1.4 **The ATLANTIC Tactical UAV.**

► **Engine type:** From micro-UAVs such as the one we have developed in this work which use electric motors to the heavier, battle ready systems using piston engines, turbofans, turboprop, push and pull or two-stroke engines, UAVs are classified by engine-type.

Lastly, UAVs are categorized in terms of the **maximum altitude** they can climb to. This is especially important in military applications to evade enemy detection, allow maximum coverage of the area under surveillance while keeping a low visibility. They could be **low altitude** micro-air vehicles with maximum ability of 1,000m; **medium altitude** UAVs with altitudes from 1000 – 10,000m and **high altitude** UAVs with flying altitudes beyond 10,000m. Typical examples include the X-45, Predator B, Global Hawk or the Darkstar UAV.

1.4 The Quad-rotor Model

Vertical take-off and landing (VTOL) vehicles are often preferred over conventional take-off and landing vehicles because of their abilities to hover over a long time, pirouette, and even side-slip. This gives VTOL vehicles the edge to be applied in aerial surveillance, ground target tracking and many other scenarios. However, rotorcrafts especially (see fig 1.5) suffer from the complexity and intricacy that comes with the complex wiring; main rotor blade

pitch cyclic control, anti-torque pitch control of the tail rotor blades and collective control of the major blade pitch (5).



4 FIGURE 1.5 THE EUROCOPTER EC-175 ROTORCRAFT [COURTESY: EADS, FIA2012]

The quadrotor often comes as a good alternative to the high-cost and complexity of the standard rotorcraft because it uses symmetrically-pitched blades which can be collectively adjusted but not individually depending on the blade's position in the rotor disc³. When the pitch and spin rate of one or more of the propellers are manipulated or altered, the vehicle motion can be controlled. This in effect changes the torque load and lift performance.

Because the torque-generated control issues were eliminative by counter-rotation, there were some quadrotors produced in the 20's and 30's but they did not work well due to poor stability and they were largely manned.

In unmanned aerial systems research lately, greater performance for the quadrotor has been found due to electronic sensors and flight control systems enhancing manoeuvrability, agility, endurance and stability. Thus, the need of the standard helicopter to be extensive and costly has been drastically reduced. Quadrotors are more robust due to the employment of the four rotors to create differential thrust; they are simple rotorcrafts with no superficial

³ Wikipedia, **Rotorcraft**

swash plates and linkages often associated with conventional rotorcraft; they use four actuators to provide six degrees of freedom (DOF) hence the colloquialism that terms them as **under-actuated systems**. This under-actuated nature compensates for complexity, power consumption and weight thus limiting translational motion in a desired direction. To overcome this, control designs are usually achieved by coupling the translational system with rotational torques in order to achieve the overall position objective (5). Essentially, they are the model upon which this research is based.

1.5 Problem Statement

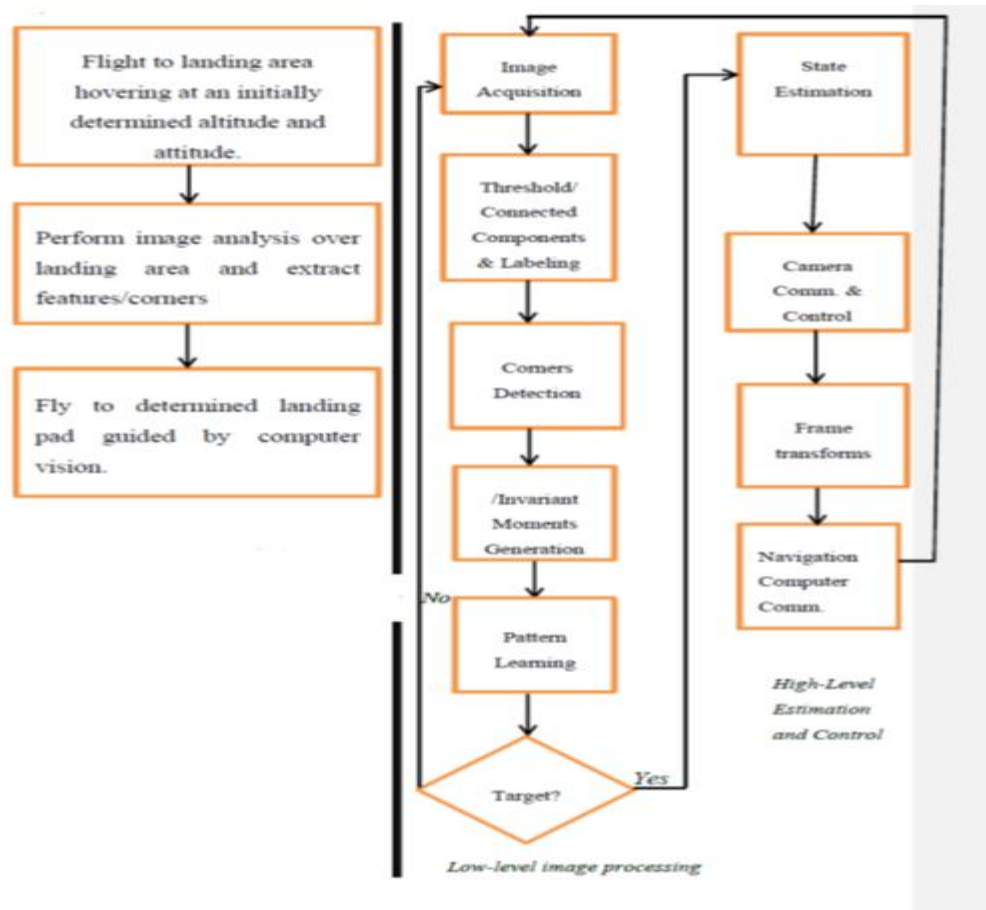
As UAVs generally use an array of sensor suite such as Global Positioning Systems (GPS) and Inertial Navigation Sensors (INS) for navigation, position and velocity estimation, there is the challenge of accurate pose estimation as these sensors only take rate measurements by which actual position and orientation are determined through integration of rate data which leads to drift in measurement over time (as a result of sensor noise). Vision-based sensing is gaining importance due to their passivity, information richness and accuracy.

During indoor flight operations, it is common for Global Positioning Systems (GPS) to fail due to blockage of a direct line of sight to the revolving GPS satellites (6). As a result, this work assumes that GPS signal is not available during the landing approach.

1.6 Research Aims

The overall objective is to develop an autonomous landing quadrotor system using machine vision for position and orientation referencing in a landing approach which could be useful in a search and rescue scenario in areas where GPS signals are occluded.

Figure 1.6 below demonstrates the task-approach to unbridling research challenges.



5. FIGURE 1.6 (LEFT) FLIGHT PATH FLOWCHART. (RIGHT) VISION SYSTEM FLOWCHART

1.7 Research Objectives

To achieve autonomous landing, first the UAV must navigate to the landing area, hold its position above the identified predetermined landing target and then fly to the target based on ego motion estimation as a GPS substitute. Therefore the objectives of this project are to

- improve existing departmental UAV for autonomous navigation purposes; navigate to a predetermined landing target, hover above

landing target during which the UAV camera acquires videos and camera frames;

- apply invariant moments, extract corners from UAV camera frames and compare results with calibrated helipad image; Learn actual helipad features if matching variance is high;
- estimate UAV pose in hovering condition; and implement simulation results on physical testbed for landing.

1.8 Dissertation Overview

The dissertation is comprised of six chapters. Chapter 2 provides discourse of existing literature, the methodologies used by different authors, testbeds, vision algorithms, controller choices and research outcomes and how they relate to the project. Chapter 3 discusses the hardware used, the low-level stability control of the UAV using a PID controller while chapter 4 emphasizes the low-level image processing feature extraction process for the vision system to be used in the controller design.

Chapter 5 presents image analysis results of simulation experiments, pattern recognition algorithm and vision-based flight tests; in Chapter 6, achievements of this research within the broader scope of the aims and objectives are enumerated. Future research directions are presented which include implementing the vision system in a control feedback loop structure for real-time image analysis and landing approach control based on ego estimation.

1.9 Project Management and Planning

In Appendix A the initial Gantt chart and scale of work progression based on project tasks are presented.

2. Literature Review

This chapter takes into perspective work that has been done on vision-based methods employed in landing and autonomous landing schemes of UAVs/helicopters and relates it to the work done in this project.

The development of an unmanned aerial vehicle capable of autonomous landing is an on-going project at the Automatic Control and Systems Engineering department of the University of Sheffield.

Prior to this work, the low-level image processing in the vision system for use on a rotary wing quadrotor has been developed by Sylvester in (7). He chooses an arbitrary landing site made of concentric ring patterns and other similitudes and performed image analyses using various segmentation methods in distinguishing a landing site from its background. He achieved consistently good results in image filtering for noise removal, connected components labeling, and edge extraction for landing targets made of concentric ring patterns. His results are however not useful for the standard H-based landing target. Other drawbacks are that a physical test-bed involving the careful selection of hardware and integration of the hardware to the system is not considered; and features in the image which are insensitive to rotation, size or translation are not considered. As the scope of this work involves estimating the UAV pose based on landing target location, Sylvester's work is limiting.

Mbaekube (8) worked on quadrotor control and flight management system development by assembling an off-the-shelf arducopter-based quadrotor UAV whose dynamics was modelled and flight simulated with the aid of a PID controller. Note his system was only stabilized within certain operating boundaries. The UAV did not fly as desired due to inaccurately selected

current and voltage, poor disturbance rejection by modelled controller and high sensitivity to attitude disturbance during hover conditions.

Therefore, this work is in two phases: first, the development of a complete flying machine capable of, remote-controlled (RC) take-off and landing, telemetry-based in-flight commands, hover at predetermined attitude and altitude with respect to target; and second, the development of the landing algorithm which includes the computer vision algorithm development and subsequent integration with the UAV system in a feedback structure for landing control.

As the bulk of this project work is focussed on vision-based autonomous landing of a rotorcraft unmanned aerial vehicle, this literature review is focussed on the computer vision research that exists in literature.

2.1 Vision-Based Landing Systems

Several works have sought to develop, implement and integrate vision systems for landing an UAV. Two broad categories are considered viz., (a) **inspection/monitoring systems** (9), (10), (11), and (12) are those that employ the vision systems in performing image processing, segmentation, feature point extraction, camera pan/tilt control, motion estimation and develop algorithms that allow for the pose estimation of the UAV with respect to the landing target – but they do not consider the vision system in a control loop architecture for autonomous landing; and (b) **visual-servoing systems** (13), (14), (15), (9), (16), (17), and (18) generally follow the approaches that the (a) categories follow but go further to integrate the computer vision into the UAV in a feedback loop structure for autonomous landing. Here, mission planning, object recognition and detection as well as pose estimation are all automated.

We shall critically look at the theories, methodologies, testbeds, research outcomes and applications of these works in the following subsections.

2.1.1 Inspection/Monitoring Systems for Landing UAVs

Including a vision sensor and within the feedback loop of UAVs is an appealing feature because otherwise inertial sensors such as global positioning systems (GPS) or inertial measurement units (IMU) are accompanied by drift over time. This is because the positional measurements are integrated over rate data leading to noise and an incorrect estimation of position or velocity. Computer vision hardware generally consume less power, are smaller in size; the computational cost of implementation is also relatively cheap for visual data processing making a strong case for their integration in a feedback loop structure.

Shakernia *et al* in (19) have done important analysis using computer vision as a feedback sensor in a control loop for landing an UAV where the vision problem was considered as an *ego-estimation problem*, that is “all feature points lie on a planar surface.” A unified geometric framework and an estimation scheme is presented for addressing the differential case. These algorithms are used to implement a computer vision in a feedback loop for state estimation to control landing. The algorithms are “linear, computationally inexpensive and capable of use in a real-time environment” but they are not physically demonstrated. Experimental results of this approach are later published in (20). The algorithm developed in (19) has been thoroughly examined using different performance criteria such as different levels of image noise, altitudes of the camera above the landing pad and then the controller is designed under a comprehensive dynamic development of the model UAV.

The vision system implemented in (9) involves a custom-made software Java-based visualization GUI showing the estimates of position and rotation of the

UAV) that extracted feature points and labels, and the camera binary image view. The testbed is a Berkeley UAV Yamaha R50 helicopter consisting of a mounted pan/tilt camera and computer box with on-board navigation and vision computers performing image processing, segmentation and feature points extraction, camera control, as well as both linear and non-linear optimization for the model-based pose estimation in real-time. The test flights of results show accuracy in the neighbourhood of 5cm and 5 degrees in the translational and rotational motion axes respectively when the UAV autonomously hovered above a small, stationary landing pad. The vision system did not affect the UAV's control while the INS/GPS measurements indicated an accuracy of 2cm. In general, the vision estimates were noisy but followed the GPS/INS measurements regardless. Overall, the vision estimates are useful enough to be placed in the control loop of hierarchical hybrid architecture of a reactive flight management system for a UAV as demonstrated by (18).

In (9) as in (19), the landing problem is treated in the ego motion estimation sense with attempts to recover the camera motion with the aid of measurements of the fixed points in the image environment by developing an algorithm specific to the planar case called the “differential” version. The linear and angular velocities of the camera are recovered using the image velocities of fixed points in a plane since when all feature points in an image are coplanar a special case of degeneracy results that makes the “8-point algorithm” ill-conditioned rendering poor estimation results and limiting the classic case of the ego estimation algorithm consequently. For mobile robots, the differential version is more appropriate for controller design as velocity estimates are necessary for controller input computation.

(19), (9) found that the most computationally intensive task for both the discrete and differential case is the linear least squares estimation of the “planar essential matrix” (matrix A see *chapter 5*) which contains all the

motion and structure parameters needed to be recovered; and the “planar essential differential matrix” (matrix B) which contains all the differential motion parameters necessary for motion and state estimation. These both require singular value decomposition (SVD) of matrices whose cost varies as the number of tracked feature points increase in turn leading to the growth of the cost function of the vision algorithm. Both found that with MATHLIB C++ library in Matlab on a 450MHz Pentium II running Linux, the vision algorithms could perform motion estimation on 25 tracked feature points at a rate of over 150Hz (19), (9).

Using performance analysis criteria such as (i) how estimation errors depend on different camera motions with respect to the observed plane, (ii) evaluation of performance in the presence of image noise in measurements, and (iii) for all simulations, comparing results with the traditional 8-point algorithm, it was established from (19), and (9)’s simulations, that (a) errors in rotational components of matrices A and B should not depend on the depths of the points; (b) as the signal to noise ratio decreased, the performance of the algorithms also decreased; (c) for the discrete and differential cases, the planar algorithm did better than the 8-point algorithm in noise sensitivity; and (d) the planar algorithm performed better than the 8-point algorithm except when the translational and optical axes are parallel.

It is worth noting that the works of (19), and (20) are based on a predefined landing site. This makes navigation to the landing site easier through the location of the landing position by predetermined characteristics and by continuously estimating the ego-motion to the target. The necessity for path planning and mission control is thus eliminated as opposed to landing on a moving target or an undefined terrain.

The works presented above are well-researched and presented but have the downside of not actually implementing the vision subsystem in a feedback control structure in landing.

2.1.2 Vision Servoing-Based Landing Control of UAVs

Visual servoing-based landing of an UAV are those approaches that fully integrate the computer vision system *with or without* other inertial sensing instruments in the landing scheme. The unique capabilities of an UAV/helicopter to hover, vertically take-off and land, perform longitudinal and lateral flight postures come at a cost: (1) they consume much power; (2) they are unstable and difficult to manoeuvre without the use of good controllers; and (3) the payload capacity is extremely limited as a result of the added costs.

Vision-based estimation enables the system from relying on external estimation devices such as satellites and beacons responsible as a resetting tool and substitute for drift-prone inertial sensors (17).

By advances in the development of inexpensive image processing chips and multi-processor architectures with high baud rates significant advances have been made in image processing. Examples include the CMUcam camera-on-board sensor series from Carnegie-Mellon University's Computer Vision Group which we will employ in this work.

In (14), (12) the design and implementation of visual landmarks in landing in a cooperative environment are presented where a fast, robust and computationally cheap algorithm is used to land a helicopter on a target composed of polygons. The camera is kept perpendicular to the ground surface in the process. Low-level image analysis was performed by thresholding the landing target after applying a 7×7 median filter for noise removal. The thresholded image was then labelled based on the 8-connectivity after which features were extracted via the first three order inertial invariant moments. These features were used for object recognition and state estimation between an offline calibrated image of the helipad and camera frames. The vision system is combined with a "low-level postural control" to

achieve precise target tracking and recognition while the helicopter updates its landing target parameters based on vision; it uses an on-board behaviour based controller to follow a path to the landing site - a “collective” controller that uses various low-level controls in different loosely-coupled partitions to achieve the overall higher-level control objective. The approach used for feature extraction in (12) has the disadvantage of lowering the effect of features that were extracted because it uses the median filter for noise removal which is a nonlinear filter that removes image features significantly.

Through the use of the ego-motion approach, the helicopter was initialized in hover at an arbitrary location from which it automatically determined a helipad, tracked it and landed the helicopter. The helicopter, during flight tests, recognized the helipad and landed to within 31cm position accuracy and 6 degrees rotational accuracy from the helipad’s centre and its primary axis respectively. Montgomery’s work is unique in that it goes beyond the works previously highlighted to demonstrate the robustness of the algorithm to find the helipad after temporarily losing it and landing on a moving target once it has stopped.

The testbed involved a gas-powered radio controlled model helicopter equipped with a PC-104 stack augmented with sensors. Boeing GMIGTS-II INS unit with a three axis gyroscope provided the state information to the computer while a Novatel RT-20 DGPS estimated the positional accuracy to within 20cm of circular error probable. A laptop served as the GCS sending high-level control commands and differential GPS corrections to the helicopter. Communication was delivered by a 2.4GHz wireless Ethernet and 1.8GHz wireless video and autonomous flight was achieved with a behaviour-based controller.

Other works such as (17) have made use of a vision system not only in landing but in a typical *simultaneous localisation and mapping* (SLAM) approach in building a map sub-system, and a navigation subsystem.

Of practical importance are the works of Hintze and O. Amidi (10), (17) who have demonstrated the complete autonomous landing of unmanned aerial vehicles. In (10) Hintze flew the vehicle first to a remote landing site in a simulation environment called RIPTIDE – a computer-based “simulation environment that makes real-time, visual, full flight envelope, pilot or operator-in-the-loop simulation readily available throughout the design cycle.” - assuming GPS data was available, then used stereo vision algorithms to generate 3D information regarding the environment from which a landing point was chosen and converted to 2D using safe landing area determination algorithms to search the 3D data earlier generated. Monocular position algorithms tracked the 2D pixel location for use in a Kalman filter that estimated the velocities and used them as inputs into the control laws to fly a normal waypoint path to the landing site. Hintze’s work is not without limitations. His work assumes: (i) a non-cooperative landing site which means when the vehicle climbs to high altitudes, the stereo ranging algorithms do not produce accurate estimates for minute objects on the ground. This is a drawback as the objects that are not captured from high altitudes by the stereo vision system could actually be dangerous during landing; (ii) Constructions such as antennas and electrical wires are difficult to distinguish from the background – a hazard for a vehicle in descent mode if the patterns are not clearly distinguished; and (iii) the inherent problem of a monocular position estimator which updates the matching template of the feature tracker after every camera frame. Wrong feedback to the template could occur due to unforeseen obstruction of the landing site.

In implementing such a design, one could use a redundant system of trackers such that in the failure of one tracker, the other ones can provide accurate information.

Amidi et al (17), (16) give a demonstration of full vision-based position estimation and landing using the Berkely Yamaha R50 platform (21), and two

Sony XC75 cameras which feed a visual odometer machine acquiring images from two independent A/D converter modules. The A/D modules provided “image digitization, real-time reconfigurable image blanking and high-speed communication ports” useful for high-speed digital processing. A real-time image convolver module, the ASIC, GEC Plessey 16488 was used for the image convolution at 10MHz per second for input image data traffic delivering 640MOPS. By taking advantage of enough features in natural scenes to lock on to arbitrary ground targets, the odometer estimates the helicopter’s lateral movements and height above the target.

This tracking algorithm implements two major execution threads. The primary thread estimates the helicopter position and the second thread measures the velocity and prepares new potential templates for future tracking. The stereo NTSC cameras are synchronised through a sync generator operating in a manner to suggest the same image field after each camera shutter opening. This guaranteed a fresh image field at every 1/60 of a second. By maintaining lock on the two image segments or templates and tracking them actively through high-speed correlation, the Odometer was able to distinguish rotational from translational motion in sensed image displacement by the aid of tagging images with helicopter attitude. Triangulation of the image displacement is thereafter used to detect the new position of the helicopter.

Results were found to be laterally- and longitudinally- accurate in position estimates to within 1.5cm despite 1-3Hz, 5-10° amplitude and attitude oscillations. There was a notable 50 -60% error in the longitudinal direction due to lower image field resolution along the direction which had impact on the template correlation positional accuracy.

2.2 Corner Detection Techniques

Among various existing segmentation methods corner detection (22), (23), (24), (25), (26) is very useful for the features extraction from a landing target

as they carry useful information regarding boundaries of curves and are good in uniqueness identification, and relative stability in images with little variance. Harris (27), Rosenfield (25), and Brady (28) have proposed corner detection methods but their approach is single-scaled and only works well if features are not many. This is a significant drawback as the landing target design chosen is such that background could be distinguished from the target itself.

Mokhtarian and He's (22), (23), (24) works were found robust and adaptable to the landing target segmentation problem. In (22), a multiscale, curvature-based shape representation method for planar curves is presented. By describing curves with different degrees of abstraction based on invariant features, results are presented that reveal evolution arc length do not alter corner boundaries. But Mokhtaran's work only uses a multiscale algorithm for localization while choosing a single-scale method for corner positions' detection. As a result, true corners are left out when deviation grows large and conversely false corners are detected when deviation is low. Global thresholding of images do not work well with this algorithm as performance of the algorithm deteriorates.

He in (23) improved upon Mokhtaran's work by instead computing the curvature of individual contours at a low level before determining corner candidates by local maxima of absolute curvature function and subsequently "comparing with an adaptive local threshold instead of a global threshold." (39). Corner candidate angles are tested to eliminate false corners based on a dynamic region of support. Results show most number of true corners were detected, false corners were greatly minimized and resulting corners closely followed reference corner lists with minimal detection of false corners.

2.3 Vision-based autonomous Landing of an Aircraft – Discussion

From sections 2.1 and 2.2 it is clear that adopting a vision system in the landing approach of an unmanned aerial vehicle is very attractive eliminating

errors associated with inertial sensing instruments. Most researchers have focussed on landing UAVs on a predetermined target or otherwise landing target whose location is not known a priori. Obviously, it is more difficult to establish a visual algorithm that can locate a safe landing target based on vision sensing instruments alone.

Based on different approaches, the estimation results for the position and orientation of the UAV/helicopter with respect to the landing target gets more accurate using the classic differential ego-motion estimation algorithm for the mobile robot. For example when Shakernia *et al* (19) demonstrated their work in (20), they found vision-based state estimates to be accurate to within 5cm and 5° along the translational and rotational axes respectively. This is because their estimation approach developed a full, dynamical model of the UAV with vision based positional sensing being self-correcting based on visual locking onto features. The motion results were used in constructing a controller. Montgomery *et al* (12), (29) used a behaviour-based controller which divided the overall controller into a set of different sub-controllers after having generated invariant moment after features segmentation and their results was only accurate to 31cm positional and 6° rotational accuracy respectively. When invariant moments are used for extracting feature points in the landing targets, however, the features tend to be more distinctive and characteristic as demonstrated by Montgomery *et al* (15, 24) where they found the accuracy between offline calibrated invariant moments and in-flight generated moments to be within a tolerance value of $\pm 10\%$.

3. Hardware and Software Requirements

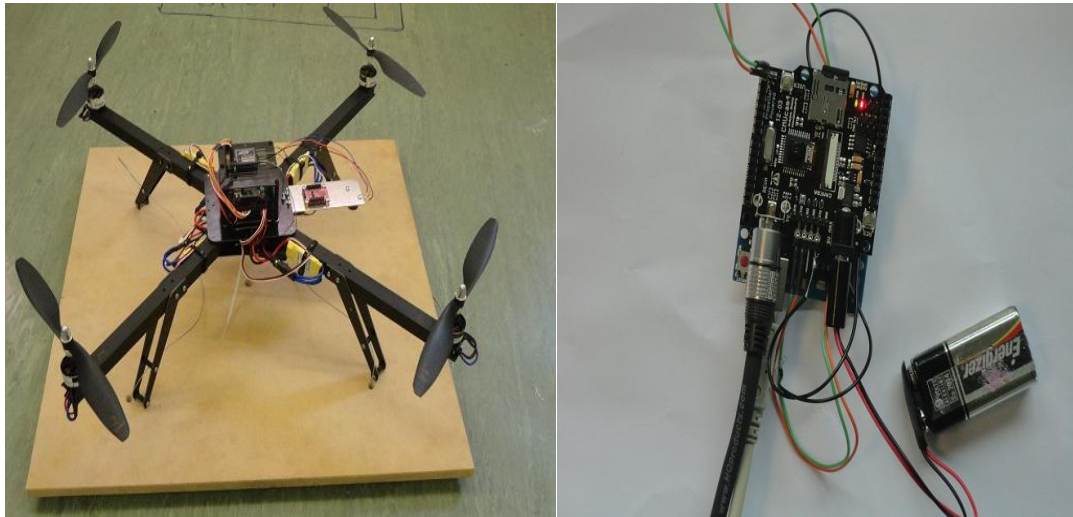
3.1 The ArduCopter Quadrotor

This project uses the arducopter quadrotor (*see figure 3.1a*) from DIY drones (30) with an on-board PID controller used in maintaining stability before the landing approach. An HP Pavilion laptop running Windows 7 is used as a GCS to send control commands via a telemetry set-up of Xbee series 2 kits to the quadrotor. The RC equipment is a Futaba 7-channel control transmitter that produces UAV angular and translational movements proportional to adjustments in transmitter stick inputs.

The vision hardware apparatus used for flight tests was the CMUcam4 camera sensor from (31) (*see fig 3.1b*) while a SONY DSC-W580 camera was used in acquiring the helipad image used as a reference. The Arducopter was chosen because of its modular structure, light frame that maintains stiffness to ensure accurate state measurement and control actuation, frame mass less than 10% of vehicular flying mass (1229g).

Low-level computation and control were performed by sending the PWM motor commands via the powerful 16MHz ATmega 2560 microcontroller. The Arducopter (30) mission planner GUI, a MAVlink-based protocol, was used to collect data, plan missions, configure radio and receiver set-up. Using a 5 cell 2200mAh Lithium polymer (LiPo) battery from Revolectrix (32), vehicle endurance was in the range 11-12 minutes though more batteries could be used to improve endurance. The battery choice was such that it was not too heavy to interfere with the maximum payload capacity of the rotorcraft. The arducopter kits come with a ready-made printed circuit board such that battery replacement and charging can be done without having to interrupt power or

device set-up. The PCB distributes power to all devices needed for system functionality.



6. FIGURE 3.1 LEFT (A) THE ARDUOPTER TESTBED IN A READY-TO-FLY MODE RIGHT (B) CMUCAM4 BOARD

The arducopter comes with an autopilot which is arduino-compatible and has capacity for six degree of freedom (DOF) IMU stabilized control (giving three-axis *attitude* and '*attitude rate*' using the in-built gyroscopes). It uses a GPS module for position hold though we are not interested in this as far as this project is concerned, magnetometer for heading determination, barometer for pressure sensing at an 'altitude hold' position, sonar sensor for automatic take-off and detection of height above ground (up to 10 meters), waypoint navigation systems.



7. Figure 3.2 THE ARDUOPTER IMU "OILPAN" SHIELD SITTING ATOP THE ARDUPILOT MEGA (APM) 2560 MICROPROCESSOR CONTROLLER (RED, BELOW)
[Credit: ArduCopter Google Code]

Thrust is generated by the *jDrones* (4) brushless AC236-358 880Kv motors controlled with pulse width modulated (PWM) **electronic speed controllers (or ESCs)** and four set of **flyer props** all from jDrones.

The IMU shield comes with a 16MB datalogging flash memory and an FTDI chip for native USB support eliminating need for a serial programming breakout board for configuration set up and mission commands.

The IMU shield and microcontroller shown in figure 3.2 are connected to a computer running Windows 7 via a USB 2.0 to mini-USB cable which recognized the FTDI chip and installed the drivers automatically. In the Device Manager, the box “set RTS on close” was checked in the port advanced settings menu of the relevant port (COM3 in our case) and the baud rate was changed to 115,200 - the default baud rate at which the GCS communicates with the computer.

3.2 Communication Development

3.2.1 Radio Set-up

The 7-channel 2.4GHz Futaba receiver (*figure 3.3b*) module was connected to the male pinouts from the end of the IMU shield. The radio in figure 3.2a has a multi-channel capacity, dial and key programming simplicity, assignable switches/functions, support for the fast advanced spread spectrum technology (FASST) making communication flawless in an environment

where there are obstacles that could downgrade signal strength.



8. FIGURE 3.3 THE FUTABA 7-CHANNEL 2.4GHZ SYSTEM(A-L)THE TRANSMITTER;(B-R)RECEIVER

Its adjustable throttle cut, aileron/elevator/rudder switches and variable rate knob also make flying and low-level control easy.

The table below (Table 3.1) shows the description of the configuration for each channel.

TABLE 3.1 CHANNELS SETUP FOR THE ARDUCOPTER

S/No	Channel	Set-Up
1	Channel 1	Aileron
2	Channel 2	Elevator
3	Channel 3	Throttle
4	Channel 4	Yaw
5	Channel 5	Mode Switch(Stabilize/Alt Hold)
6	Channel 6	Camera
7	Channel 7	AUX (Unused)

3.2.2 Telemetry Setup

Using the Zigbee mesh firmware technology eliminated the need for USB wire-in-the-loop such that we were able to achieve remote connectivity and data-logging albeit at a lower baud rate baud – 57600bps.



9 FIGURE 3.4 DIGIMESH'S XBEE SERIES 2 ROUTER AND COORDINATOR MODULES

Zigbee was favoured for its powerful data routing capabilities (2mW for the Xbee S2 modules), range extension of up to 133ft (allowing hopping of data from node to node), improved reliability through self-healing and its open standard technology allowing for interoperability with devices from different vendors.

The **Xbee S2** from Digi was chosen because of its (1) low current draw of 40mA at 3.3V for both the Tx and Rx modules (2) small form factor hence minimizing overall weight of the UAV, (3) extensive command set (4) self-routing, self-healing and fault-tolerant mesh networking (5) point-to-point, point-to-multipoint and peer-to-peer topologies (6) direct spread spectrum technology and (7) retries and acknowledgments. The downside of the Xbee S2 technology is its complexity of configuration for implementation for the project's purpose.

To set up, the AT command module was used to set one of the device pairs as a router with a USB connection to a laptop running Windows 7 at the default baud rate of 9,600bps while the other pair was configured as a

coordinator end device sitting on a Sparkfun Explorer mounting board receptacle. The coordinator was wired to the IMU shield via the extended serial ports from the oilpan.

Using the transparent mode, the DH (Destination Address High) and DL (Destination Address Low) parameters of the source node were matched to the 64-bit SH (Serial High) and SL (Serial Low) address of the destination node. The Zigbee protocol is dependent on 16-bit network addressing scheme for routing and thus the 64-bit addresses are converted to 16-bit network address before data transmission. When a module has no information as to the 16-bit network address of a given 64-bit address, it transmits a broadcast network address discovery command which a module with a corresponding 64-bit address responds to by transmitting its 16-bit network address back. Data only gets transmitted when the network address discovery is established.

3.2.3 Sonar Mount

For low altitude ranging and obstacle avoidance, Maxbotics' LV-EZ0 Sonar Sensor was mounted about three-half inches from the UAV body to avoid electrical noise from the speed controllers and other electronics. The sonar supports low level altitude hold at heights below 10 meters. Above 10 meters, the barometric sensor takes over altitude sensing. Operation was verified by measuring the voltage between the AN-terminal and ground after connecting the battery. The voltage was found to be 10mV indicating conformance to factory calibration.

3.2.4 Ground Station Set-up and Configuration

The arducopter supports industrial standard MAVlink communication protocol which can efficiently pack C-structs over serial channels while relaying the packets to the ground station. It served as the main communication tool for the ArduPilotMega.

The Mission Planner software by Michael Osborne (30) was used as the GCS to collect flight datalogs, tune the PIDs for stability control, remote take-off and hover and landing configuration.



10. FIG 3.5 THE MISSION PLANNER GUI SOFTWARE

The mission planner supports python scripting and the in-built script for mid-air roll was modified for take-off, navigation to the landing target based on expert knowledge and hovering at a preselected altitude while camera acquired images and video streaming of the landing target. Using the RCA 3.5mm jack connector on board the camera sensor, video of landing target was acquired using a USB video capture device.

The acquired image gets dumped to the on-board μSD -card via commands to the camera from the parallax serial terminal interface which were later collected for image analysis (see *chapter 4*). The sonar ranger was enabled in the mission planner hardware configuration tab to integrate the Sonar device with the system.

3.3 Electronic Speed Controllers (ESCs) and Propeller configuration

The arducopter is designed in such a way that it can only fly correctly if all four ESCs are set-up to respond simultaneously to the PWM and RC system. This process is called 'arming' the motors. In order to be able to arm the controllers, the ESCs need to be calibrated and props correctly set up.

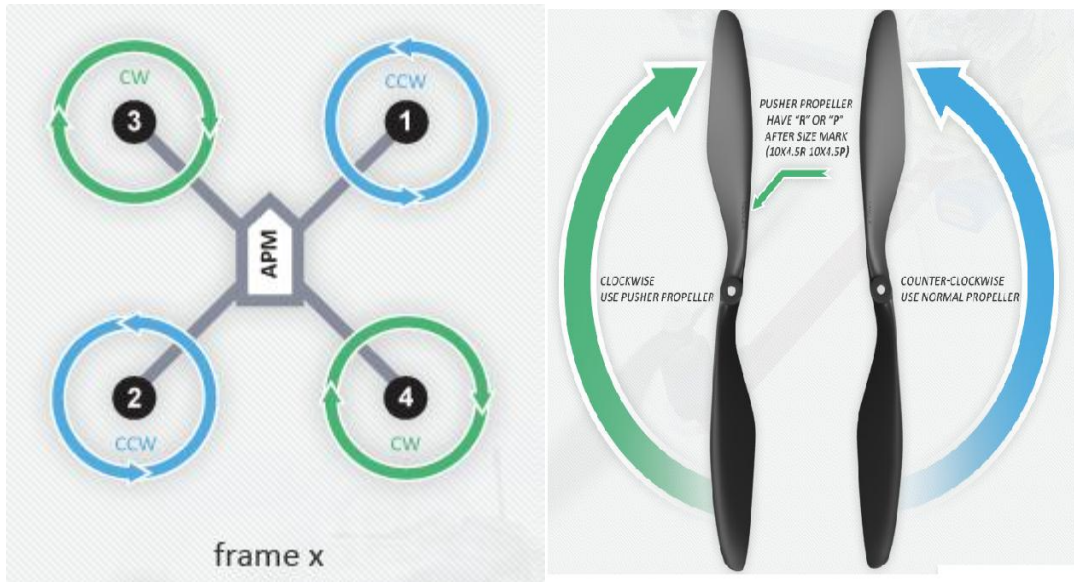
3.3.1 ESC's Calibration

To calibrate, the automatic calibration method was followed (30) which included first, disconnecting the USB; second, putting the throttle high and connecting the LiPo battery to power the APM; third, booting the APM and then disconnecting the LiPo before it is subsequently reconnected. This calibrated the ESCs by sending high PWM to the four ESCs at once. The throttle stick was then dropped to the lowest position with an arming beep confirmation.

By setting the transmitter to the "Stabilize Mode" earlier configured in the mission planner in order to ensure safety in the event of an accident during mission testing, flight tests were performed with the motors spinning at the same speed and starting at the same time.

3.3.2 Props Set-up

Pusher and puller props were acquired from DIYdrones (30) and the arducopter was configured in an *X* –setup (figure 3.6a). The pusher props spin clockwise while the puller props spin anticlockwise. Together, they gave the arducopter quadrotor the lift for flight in after power-up.

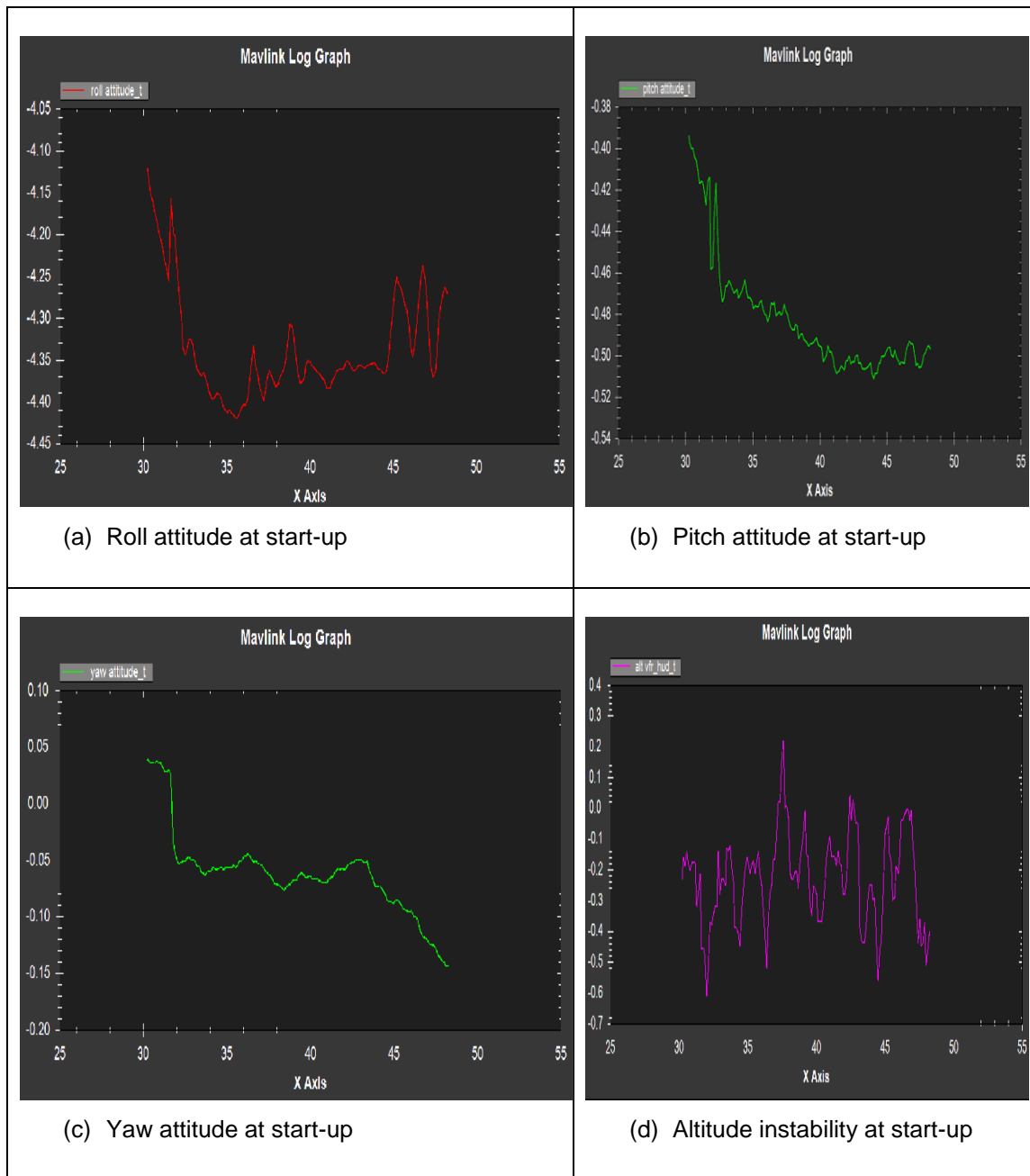


11. FIGURE 3.6 LEFT (A) THE ARDUOPTER IN A 'FRAME X' FORMAT; RIGHT (B) THE PUSHER AND PULLER PROPS DIRECTION

3.4 Results of Flight Tests

Having configured the arducopter, on power-up and elevating the throttle stick on the Futaba transmitter, based on signal pwm outputs from the calibrated causing the UAV to fly. Recorded measurements are shown in figures 3.7 and 3.8 respectively.

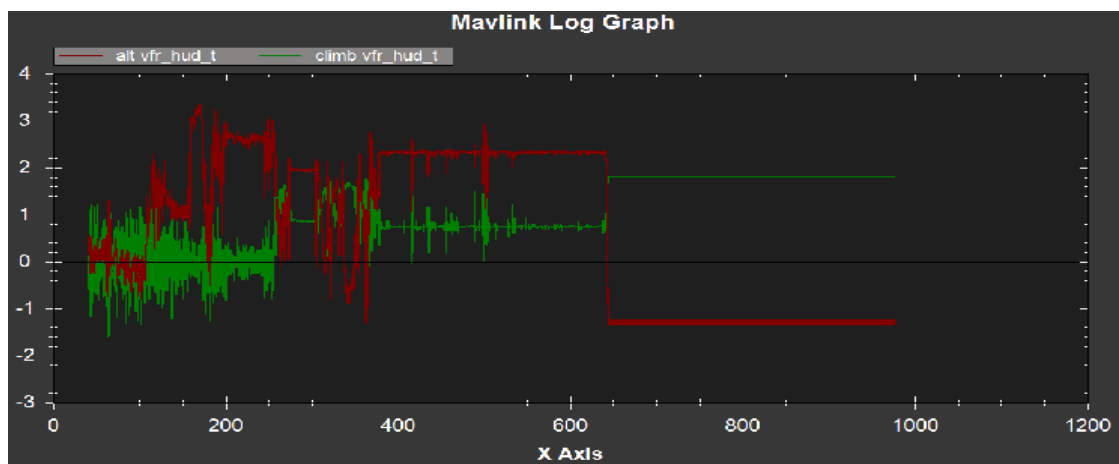
Notice attitude instability in roll, yaw and pitch directions in figure 3.7 a – c and altitude gyrations in figure 3.7d. The arducopter comes a PID controller which was used to improve performance. By varying the proportional, P and derivative, D, components of the PID stability was achieved to the extent that the UAV was able to move freely in the absence of overwhelming disturbance.



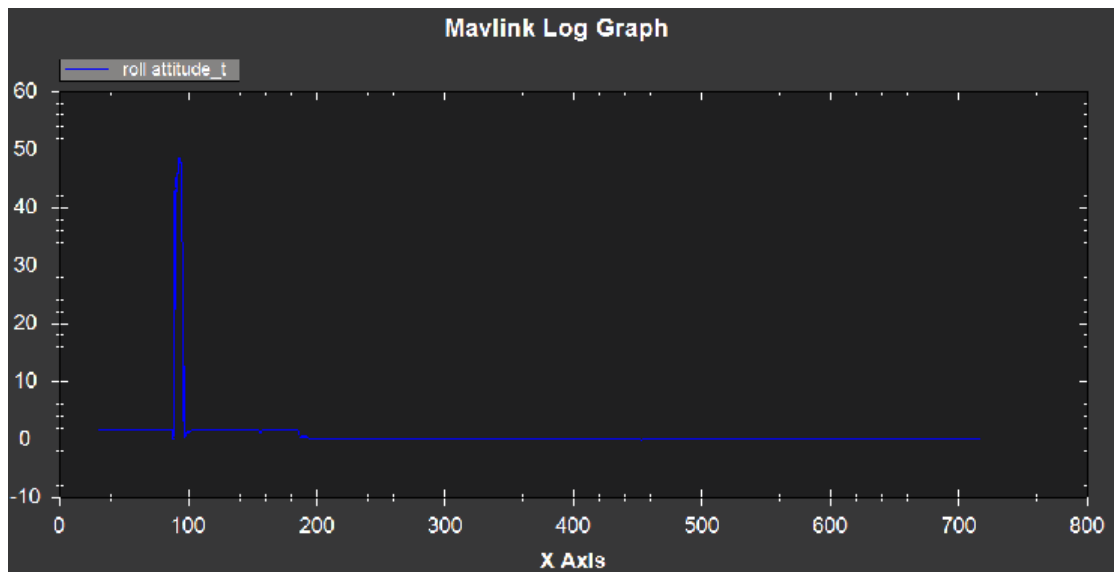
12. **FIGURE 3.7 AIRCRAFT INSTABILITY AT INITIAL START-UP**

For normal flights, only the proportional term in the respective PID controllers were adjusted in stabilize mode. Figure 3.8a below shows the behaviour of the UAV in an altitude hold mode (by decreasing the constant gain term, acceleration was fed back into the controller such that barometric perturbations were compensated for. The slight deviations from settling behaviour observed is due to the ground force reaction

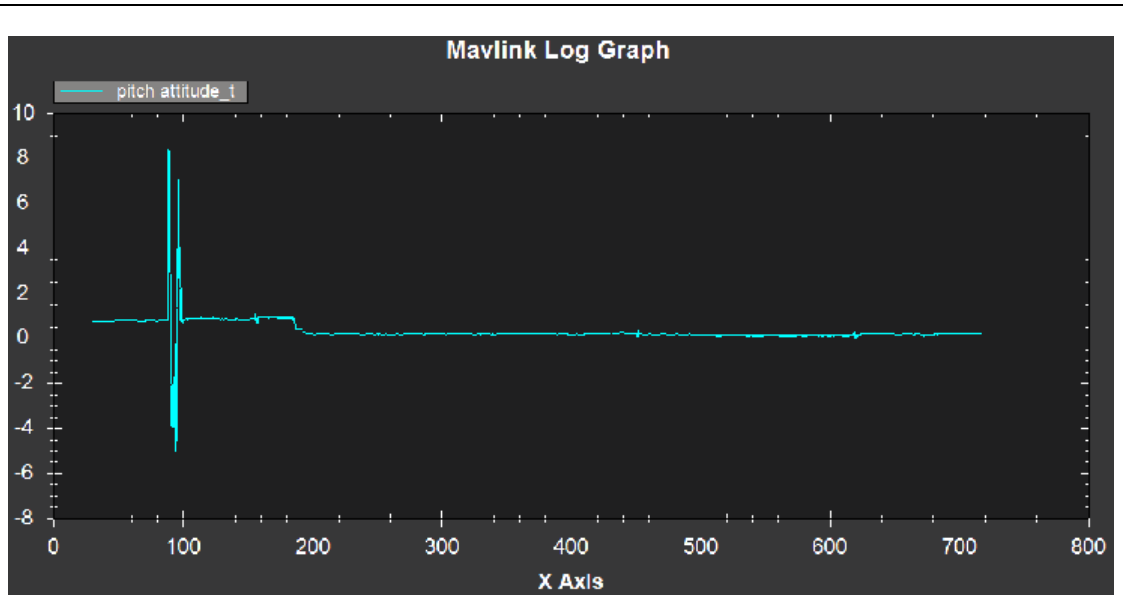
to the UAV as they affect the altitude stability of the aircraft at low heights. Overall, it settles to a constant value allowing for hover conditions



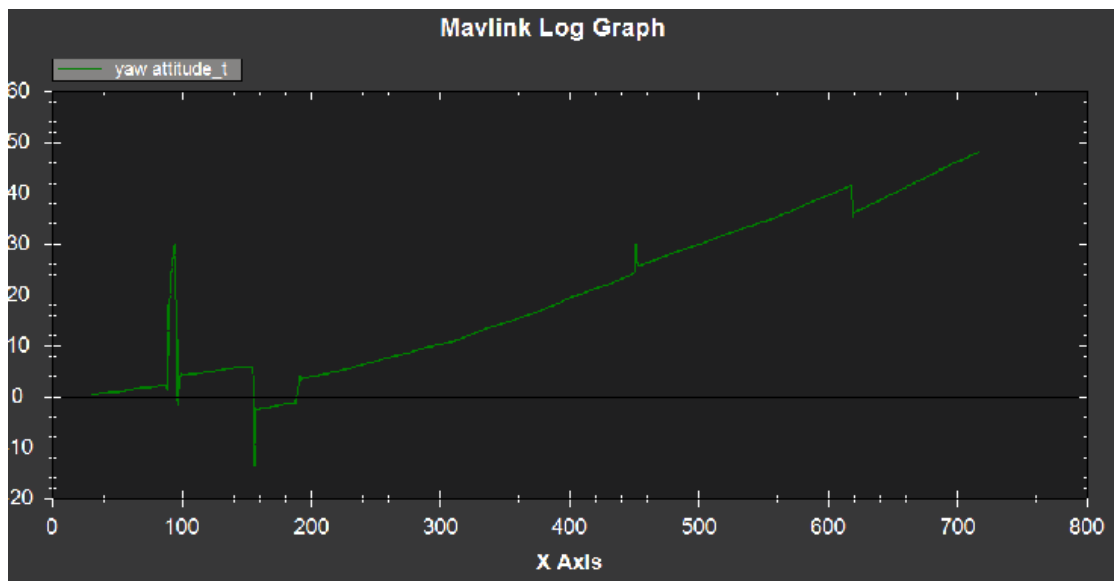
Altitude and Climb Visualizations in PID after-tuning



(b) Roll attitude after PID tuning



(b) Pitch attitude after PID tuning



(c) Graph demonstrating relative stability in the yaw direction

13 FIGURE 3.8 ATTITUDE AND ALTITUDE BEHAVIOUR AFTER TUNING EXPERIMENT

Figures 3.8b-c show flight results after tuning roll, pitch, and yaw PD-controllers. The arducopter's functionality is such that when in hover condition based on mode switch to the "Alt Hold" position (see (30) for details), when the RC-controller stick is moved in a desired direction, the previous mode is exited. Figure 3.8c demonstrates this characteristic based on moving the stick

in the yaw direction in “Alt Hold” position mode. Here the arducopter yaws right.

3.5 Camera Set-up and Incorporation

An off-the-shelf camera module sensor was obtained from (31) (see *fig 3.1.1b*) called the **CMUcam4**. It has an on-board parallax P8X32A propeller multi-processing chip capable of object recognition tasks based on an in-built interpreter which supports high-level object oriented programming (much similar to C called **SPIN**) and low-level assembly language. It performs high-speed embedded processing with little power- and current-draw. Connected to a low-voltage CMOS OV9665 image sensor via a serial camera control bus (SCCB) interface, it provides sub-sampled, windowed 8- and 10-bit images in RGB565/YUV655 colour formats with a VGA resolution of 640×480 at a baud rate of 30 frames per second (fps); images are dumped to a 4GB μ SD card over serial at $(640:320:160:80) \times (480:240:120:60)$ resolution; it provides mean, median, mode and standard deviation data collection sampled from a 40×20 resolution and colour tracking via threshold segmentation of images, windowing and histogram generation of up to 128 bins.

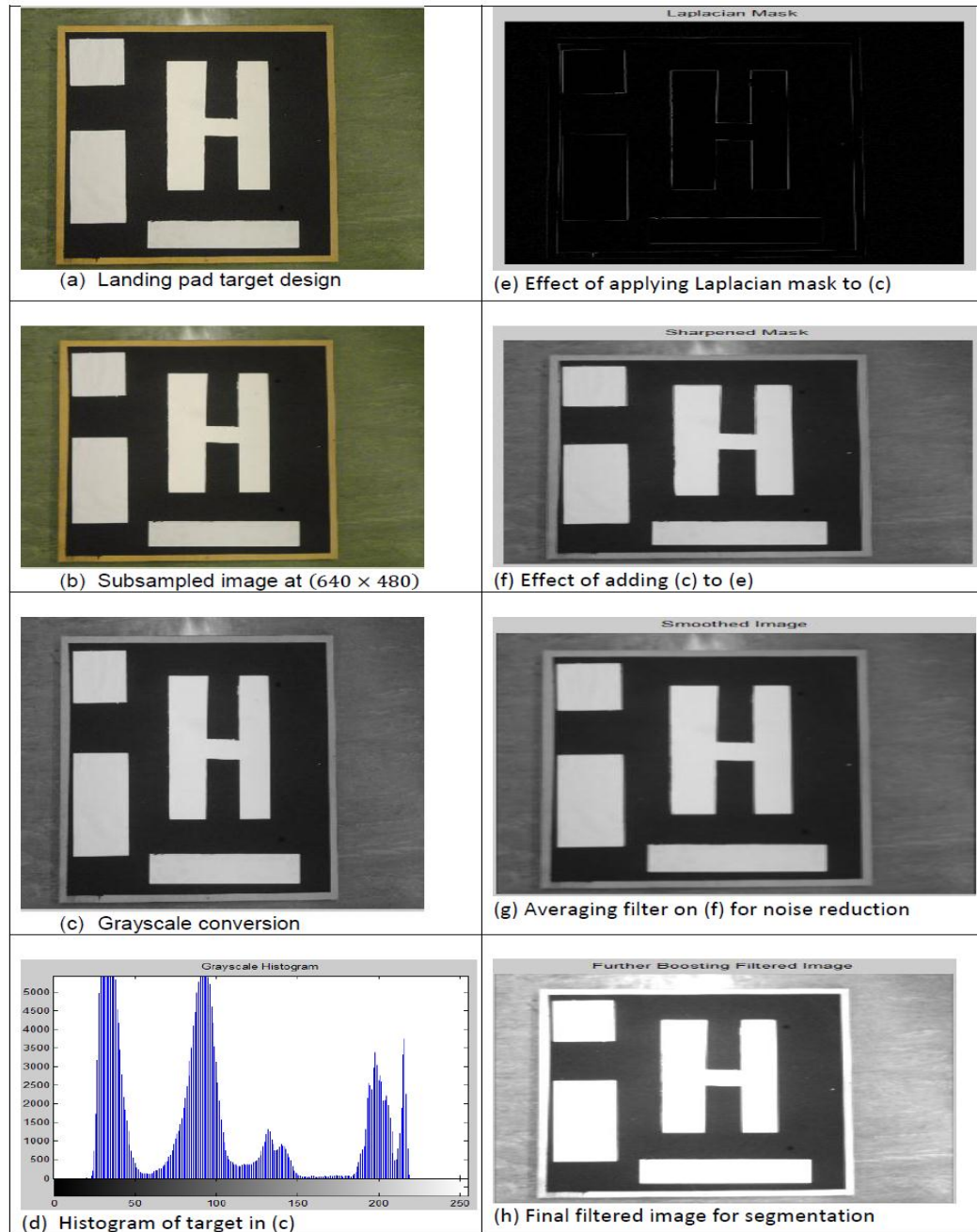
4. Visual Sensing Algorithm

The sections below present the simulation of a simple model of the vision algorithm. The goal is to locate the helipad target, detect corners, extract invariant feature points from the detected corners and then develop a pattern recognition model for the CMUcam4 sensor based on generated invariants. The process therefore includes: (1) thresholding the landing target grayscale image offline (2) segmenting the background from the target, (3) corner detection and corner pairs labelling, and (4) first, second and third invariant moments generation from the detected corners (5) an object recognition algorithm for the CMUcam4 based on generated invariants.

4.1 Low-level Image Pre-Processing

Pre-processing involves careful landing target design selection that simplifies target segmentation from background such that distinct invariant features can be further extracted. Using a 55cm^2 rectangular wooden board, the background was painted black and white rectangles were included at the sides with an 'H' helipad symbol inserted in the middle (see *figure 4.1a*). The colour contrast between the landing target and background made for simple features extraction as no other white region encircled the 'H' symbol totally. Corner detection feature points were chosen because of their robustness, high density of feature points over pixel area and low computational cost. In particular, 4-sided figures were used for the landing target because of their well-defined corners with sharp angles such that true corners could be differentiated from false corners and because corner merits get maximized under perspective projection and pixel quantization (20).

During test flights, colour tracking by the CMUcam4 was exploited but the high sensitivity of the OmniVision 9665 CMOS camera sensor to sunlight prevented tracking.



14 **FIGURE 4.1: DESIGN OF LANDING TARGET AND RESULTS OF DIFFERENT IMAGE ANALYSIS PROCEDURES**

4.1.1 Thresholding

Image analysis procedures are generally based on one of two methods: segmenting based on discontinuities or similarities in images (33). Discontinuity algorithms involve dividing an image into sub-regions based on sudden changes in intensity (among these are edge-, point-, and line-detection methods), while similarity algorithms involve using pre-existing definitions (such as thresholding or regions of interest extraction) to segment an image.

As speed in target identification is a crucial factor in deciding the UAV pose estimation by the robot, thresholding techniques are best suitable for the overall objective. Prior to thresholding, the acquired image needs to be subsampled, filtered (or blurred for noise removal) and finally digitized through an appropriate threshold level.

The image in figure 4.1a was acquired offline with a Sony DSC-W580 camera at 72 Pixel per inch and 2592×1944 resolution. To allow for easy matching with the CMUcam4¹, it was first subsampled to a size of 640×480 using the MATLAB Image Processing Toolbox (*see code snippet below*)

```
% Read Images from a Sony DSC SD-card dumped images
clear all; close all; clc;
cd('C:\Users\Olalekan\Desktop\feature_extraction\feature_extraction\feature_extraction') J = imread('helipad_1.jpg'); % Original Image at 2592 x 1944
a = imresize(J,[640 480], 'bilinear'); imshow(J) %Subsample Image
```

Using the *bilinear interpolation* algorithm the original image ($1944 \times 2592 \times 3$) was shrunk to a (640×480) resolution with the 'imresize' command in Matlab. The bilinear interpolation uses the 4-nearest pixel values located in the diagonal of each pixel to find the appropriate colour intensity of a pixel

¹ The CMUcam4 outputs (640×480) image dumps over serial

through row and column deletion. The interpolation applies the following relation to obtain the desired gray level for each new pixel it shrinks:

$$f(x, y) = a_1 + a_2x + a_3y + a_4y \quad (4.1)$$

where corner values are known and used to uniquely determine bilinear interpolation coefficients a_1 , a_2 , a_3 , and a_4 (37).

One of the *benefits* of the *imresize* function in the matlab image toolbox is its use of antialiasing to limit the high-frequency components of the original image by blurring the image before sampling (35,36) thereby minimizing the effect of *Moiré Patterns*².

For most excellent machine interpretation of results, the subsampled image was converted from RGB to grayscale representation by removing the hue and saturation components but preserving luminance using the following relation (35):

$$U = 0.299 \times R + 0.596 \times G + 0.211 \times B \quad (4.2)$$

Where U is the 2-dimensional scaled version of the original image. R, G , and B in (4.2) represent the red, green and blue components in the original image respectively. Performing the conversion in (4.2) made the proposed algorithm computationally cost-efficient and adaptable on the CMUcam4 processing platform. The 'rgb2gray' command was used to

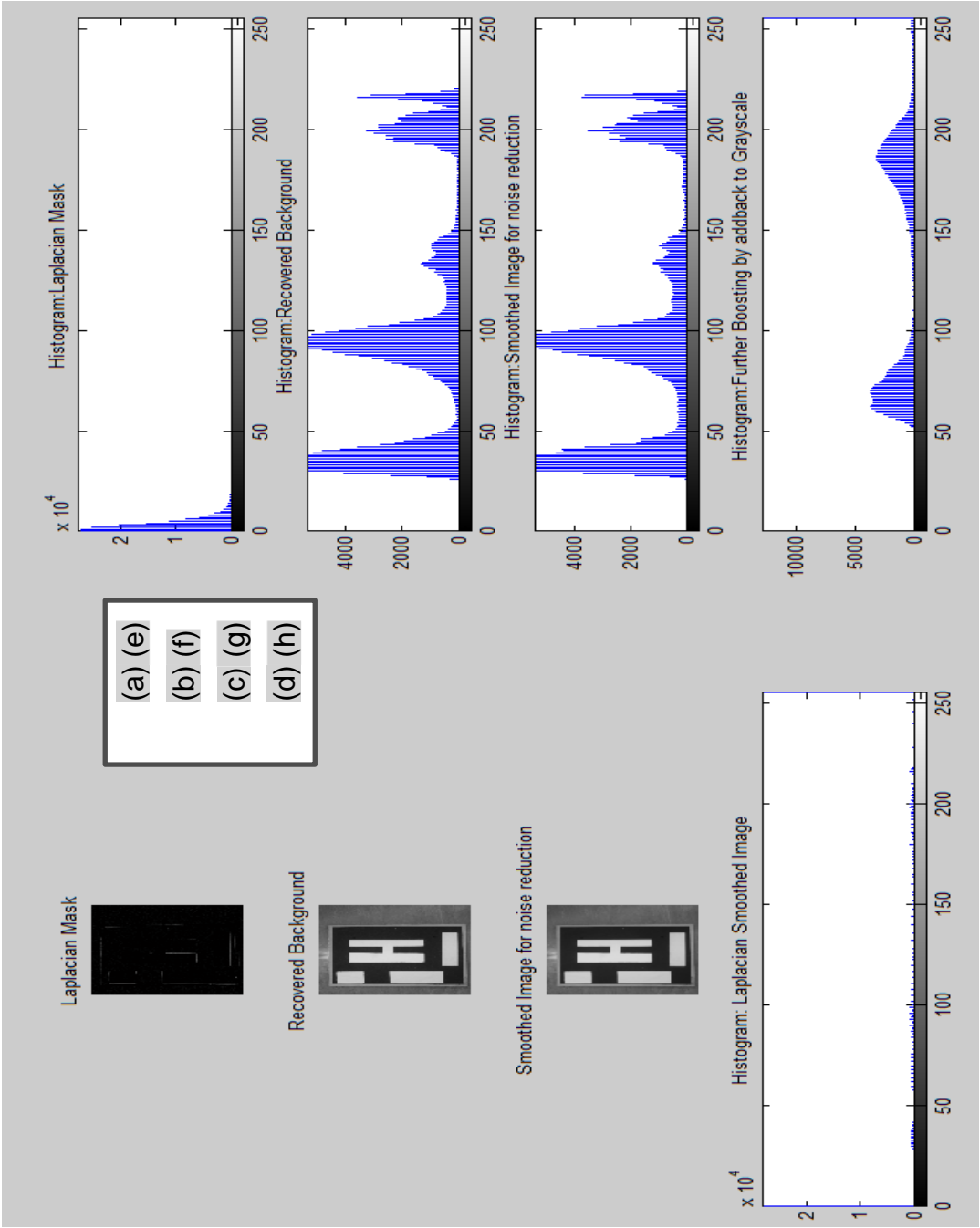
```
%% RGB to Grayscale Colour Conversion
a_gray = rgb2gray(a); % Subsampled rgb coloured image is
turned to gray
```

perform equation (4.2) as shown in the snippet above.

In figure (4.1d), the grayscale histogram of the landing target shows the five different peaks (measured along the vertical axis) unevenly contrasted across

² Moiré Patterns are the effect of aliased frequencies due to the division of image periodicity.

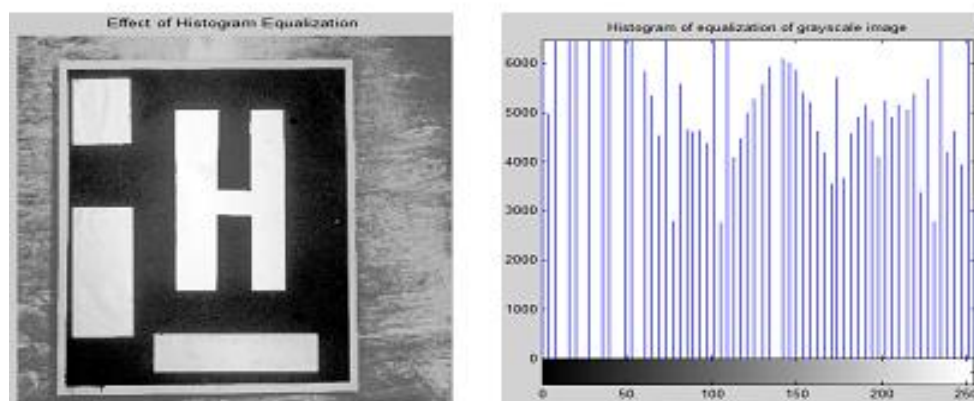
gray levels (horizontal axis) with high number of pixels concentrated on the low(dark) side of the gray scale. This means applying the segmentation algorithm would produce a poor distinction between the landing target and background.



15 FIGURE 4.21 SPATIAL ENHANCEMENTS FOR IMAGE SEGMENTATION.

To overcome this, a way needs to be found to make the pixels occupy the entire range of possible gray levels and be uniformly distributed such that the resulting image will have a high contrast appearance and exhibit a reflection of all gray tones.

Median filter is a popular approach towards overcoming this difficulty but it is a nonlinear approach that removes image features. Image equalization, while effective, did not have the effect of the combined method in figure 4.21 due to its dark gray-level distributions and lack of uniformity (*fig 4.22*)



16. FIGURE 4.22 EFFECT OF APPLYING THE HISTOGRAM EQUALIZATION METHOD ON GRAYSCALE IMAGE OF FIG 4.1C

Different spatial enhancement methods were combined (*see results in fig. 4.2*) using mask generated from a smoothed Laplacian gradient of the grayscale image.

The level of noise in the grayscale image of *fig 4.1c* needs to be further reduced. One of the ways of removing noise levels is by applying a smoothing filter to average gray levels in neighbouring pixels and replace each pixel with the generated average. But this method has the disadvantage of blurring edges (33) as a result of sharp transitions in pixel values and therefore does not lend itself to the direction to be taken.

The Laplacian (∇^2) mask finds usefulness in this application because of its derivative nature on an image $f(x,y)$ (see *equation 4.3*), by highlighting discontinuities in gray level while playing down regions with little difference in gray levels.

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \quad (4.3)$$

The fact that it uses a second derivative operation makes it more ideal because it can process finer details at the expense of a noisy output image.

Applying the Laplacian to the grayscale image produced an image with a dark background and gray edges (*fig 4.1e*). The actual image was then recovered by adding the original grayscale image back to the Laplacian obtained. The result of this operation is shown in *fig 4.1f* where all the background has been recovered through matrix addition. It is worth noting that addition was performed in this instance because the centre of the Laplacian used was positive (see *snippet code below*). If otherwise, element-wise subtraction would have been performed.

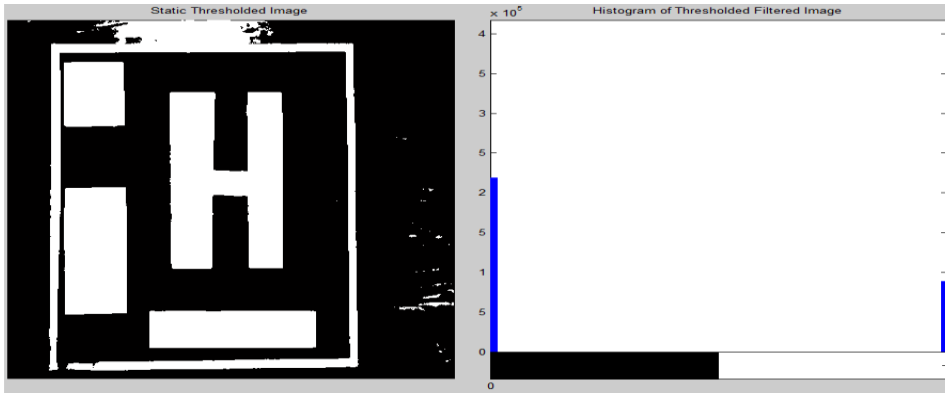
```
H_lap      = fspecial('laplacian'); %Generate the 3 x 3 laplacian
filter
a_lap      = imfilter(a_gray, H_lap); % Apply the Laplacian on
the Grayscale Image
a_sharp    = a_gray + a_lap; % Recover background in Laplacian by
adding back to Grayscale Image
H_ave      = fspecial('average'); %Generate 3x3 avg filter to
smoothen gradient of laplacian image
a_smoothed = imfilter(a_sharp, H_ave); %Smooth obtained gradient
a_lap_prod = a_smoothed.*a_lap; %Combine best features of Lap and
Smooth
a_sharp_boost = a_gray + a_smoothed ;    % Further boost smoothing
% figure()    ; imhist(a_gray); title ('Grayscale
Histogram');
```

The histogram of the recovered image is shown in *fig 4.2f* with the label 'Histogram: Recovered Background'. It can be seen that the histogram is spread across the whole gray region but is marked by abrupt transitions to other gray levels. This generated a noise (though lower than the Laplacian's but still significant) that was lowered by the 3×3 averaging mask shown in

the code snippet above. The resulting product was multiplied by the Laplacian mask and then added back to the grayscale image to obtain the final filtered image.

This procedure took the strengths of the Laplacian and the smoothing gradient to deliver the best features in the final filtered image (*fig 4.1h*); the histogram plot of the final filtered image is shown in *fig 4.2h* displaying a balanced, well-contrasted and evenly distributed gray-levels in the overall image that will be useful in segmenting the image into features that are useful for extraction. Bottom-left of *figure 4.2* reveals the dominance of strong edges (scaled down by 10^4) and the absence of visible noise

Having filtered the image, it was segmented based on global thresholding such that the background could be clearly distinguished from the white rectangles/H design. Sylvester in [13] found after several landing target design tests that a threshold value of 85% was optimal for designs such as *figure 4.1c* and his recommendation has been adopted.



17 FIGURE 4.3 LEFT (A) THRESHOLDED IMAGE RIGHT (B) HISTOGRAM OF IMAGE (A)

Global thresholding partitions the image into regions of black or white based on equation (4.4) and generates an output $h(x, y)$:

$$h(x, y) = \begin{cases} 1 & \text{if } f(x, y) > T \\ 0 & \text{if } f(x, y) \leq T \end{cases} \quad (4.4)$$

where $f(x, y)$ represents the input image (in this case fig 4.3a) and T is the global threshold value ($T = 85\%$). Equation (4.4) has the significance that pixels labelled 1 reflect objects we are interested in while those labelled 0 correspond to the background.

The result of applying this threshold is presented in figure 4.3b showing the desired outcome was achieved with two pixel intensities packed to either side of the gray levels (black and white respectively).

```
% Thresholding
level      = 0.85; %Threshold value
BW         = im2bw(a_sharp_boost, level); %Binarize image
          according to chosen threshold
```

4.1.2 Segmentation and Corner Detection Algorithm

Objects other than the helipad and boxes need to be filtered out. This involves segmenting the thresholded image of figure 4.3a into regions of desired objects and background.

Segmentation partitions the image R into regions of interest R_1, R_2, \dots, R_n such that:

- (i) $R_1 \cup R_2 \cup \dots \cup R_n = R$
- (ii) R_x is a connected region for all x
- (iii) $R_x \cap R_y = \emptyset$ for all x and $y, x \neq y$
- (iv) $f(R_x) = \text{TRUE}$ for $x = 1, 2, \dots, n$.
- (v) $f(R_x \cup R_y) = \text{FALSE}$ for $x \neq y$.

where $f(R_x)$ is a classifier for the region R_x and \emptyset is an empty set.

Interpreted, condition (i) implies every pixel must be covered in a region, (ii) implies connectivity of points in a region and (iii) means there must be no intersection of extracted regions; (iv) defines necessary conditions for pixels in a region while (v) indicates R_x and R_y are different with regards to the classifier f .

Popular corner detection algorithms include those proposed by Harris and Rosenfield (27), (25) but they are single scaled and when applied to images of big sizes, they fail. The curvature scale space technique (23), (24) increases the number of true corners detected; reduces the quantity of false corners; achieves consistent detection performance from image to image and identifies corners based on local and global curvatures detecting paramount features of various sizes and neglecting the trivial details, thus it is very appropriate for recovering invariant geometric patterns of multiple scales such as we are presented with in this work.

He and Yung in (23), (24) improves the corner extraction method proposed by Mokhtarian in (22) by computing curvatures at a “fixed low scale for each contour to retain all true corners” before local maxima and *false corners* are considered as corner points. Their approach is used in fig. 4.3a for the intended corner segmentation.

First, the definition of a curvature:

$$K(u, \sigma) = \frac{X_u(u, \sigma)Y_{u,u}(u, \sigma) - X_{uu}(u, \sigma)Y_u(u, \sigma)}{(X_u(u, \sigma)^2 + Y_u(u, \sigma)^2)^{3/2}} \quad (4.5)$$

where $X_u(u, \sigma) = \frac{\partial}{\partial u}(x(u) \circledast g(u, \sigma)) = x(u) \circledast g_u(u, \sigma)$,

$$X_{uu}(u, \sigma) = \frac{\partial^2}{\partial u^2}(x(u) \circledast g(u, \sigma)) = x(u) \circledast g_{uu}(u, \sigma)$$

$Y_u(u, \sigma) = y(u) \circledast g_u(u, \sigma)$ and $Y_{uu}(u, \sigma) = y(u) \circledast g_{uu}(u, \sigma)$.

\circledast is the convolution operator, $g(u, \sigma)$ is the Gaussian function with standard deviation or width σ and x and y are points on the planar curve of interest.

The algorithm derives a binary edge map using the canny edge detector, after which it produces the contours of the edges from the resulting edge map filling the gaps in the contours and identifying T-junctions. Curvature is then

computed at a low scale for each contour to retain all true corners while regarding local maxima for absolute curvatures as initial corner candidates.

The adaptive curvature threshold (see *equation 4.6*) is further used to remove round corners from the initial candidates before false corners arising from quantized noise and unimportant details are dispelled through the evaluation of the angles of corner candidates in a dynamic *support region*³.

If $T(u)$ is the adaptive threshold, then it is defined:

$$T(u) = C \times \bar{K} = 1.5 \times \frac{1}{L_1 + L_2 + 1} \sum_{i=u-L_2}^{u+L_1} K(i) \quad (4.6)$$

where \bar{K} is the mean of the neighbourhood region, u is the corner point position and L_1, L_2 are the sizes of the region of support (ROS) defined as the region of a neighbouring local curvature minima to the next minima for a curvature decreasing from a candidate point at both ends. C is a scaling coefficient and 1.5 was chosen because it works well for a lot of images.

Basing the corner criterion on the conditions:

If $160^\circ \leq \text{angle of } C_i \leq 200^\circ$, then C_i is a **False Corner**
 Else, C_i is a **True Corner**.

where angle of C_i is defined as $\angle C_i = |\tan^{-1}(\nabla Y1 / \nabla X1) - \tan^{-1}(\nabla Y2 / \nabla X2)|$ and

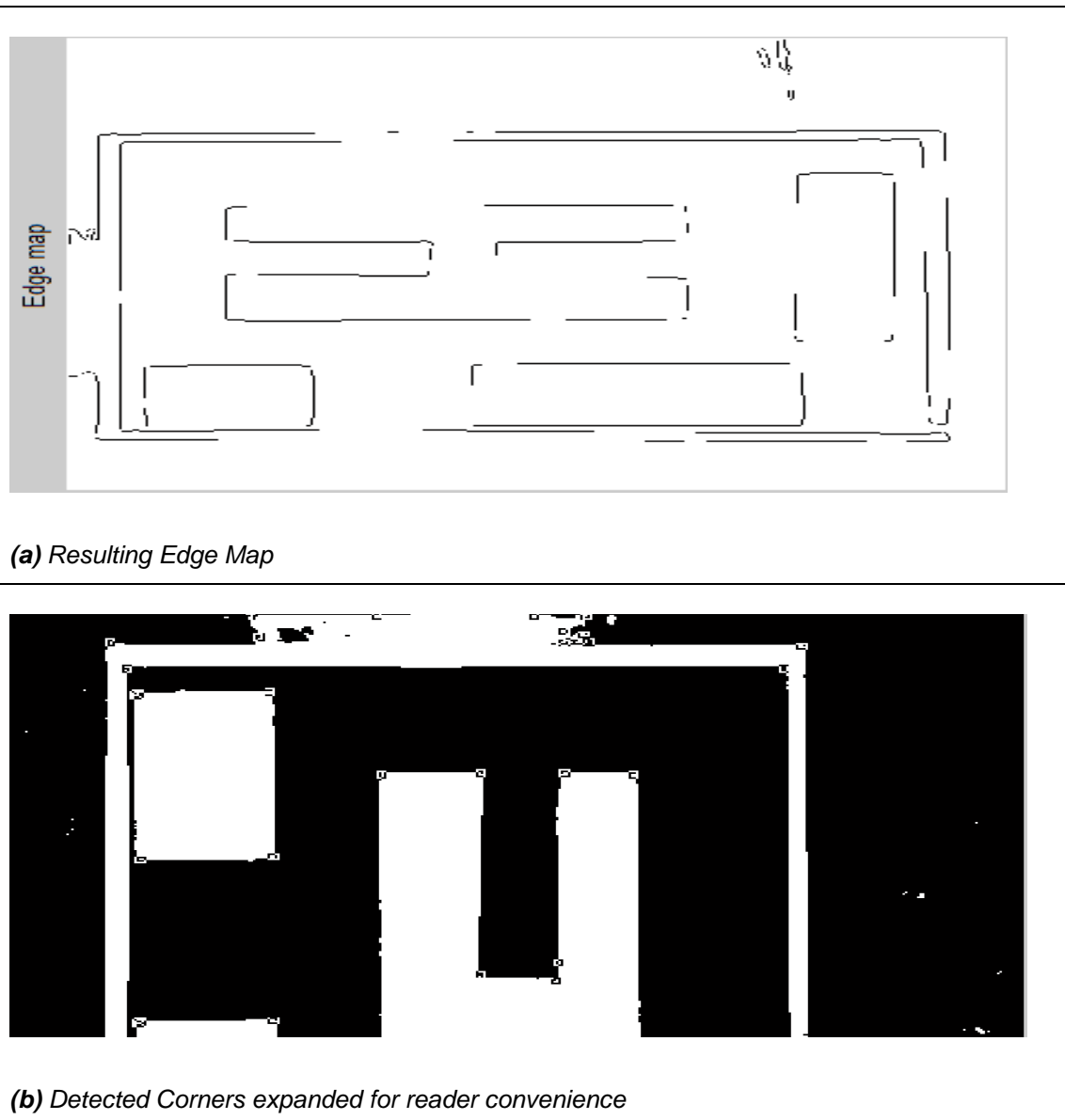
$$\nabla X1 = \frac{1}{L_1} \sum_{i=u+1}^{u+L_1} X(i) - X(u); \text{ and } \nabla Y1 = \frac{1}{L_1} \sum_{i=u+1}^{u+L_1} Y(i) - Y(u); \quad (4.7a)$$

$$\nabla X2 = \frac{1}{L_2} \sum_{i=u-L_2}^{u-1} X(i) - X(u); \text{ and } \nabla Y2 = \frac{1}{L_2} \sum_{i=u-L_2}^{u-1} Y(i) - Y(u); \quad (4.7b)$$

The landing pad was able to be segmented according to predefined criterion.

³ A support region is the region from a neighbouring local curvature minimum to the next when a curvature is perfectly decreasing from the candidate point at both ends.

Initial tests with the test data showed the algorithm was tolerant enough to separate between landing target and other objects present in the image (equations 4.5, 4.6, and 4.7).



18. FIGURE 4.4 RESULT OF APPLYING EQUATIONS (4.5) AND (4.6) TO FIG (4.3A).

We see from figure 4.4b that the corners are well-marked, rectangles are correctly labelled and there are little false corners. More results with test images are shown in Appendix C. The edgeness of the helipad is however lost at some positions as shown in figure 4.4a. We are not so much concerned

with this as the corner pairs are the chief features upon which correlation between UAV frames and helipad image are based.

During actual flights, the thresholding and corner extraction algorithms will be applied to each CMUcam4 frame and positional pairs of corners are compared to the calibrated helipad image shown above. Results of different test images are presented in Appendix C.

4.2 Features Detection

The 2-D image in figure 4.4b contains definite features about the image whose various properties such as area, perimeter, region, bounded blobs, convexity, extrema or moments can be used to extract properties insensitive to rotation, translation and positional attributes. It is well known that moments are invariant properties that carry distinct information particular to an image.

In order to describe a feature vector that is independent of rotation, size or position in the visual field, the two-dimensional central moment invariants are used for the feature description [(14), (29), (34), and (35)].

A 2D continuous function can be denoted by the Cauchy-Riemann density distributed function $g(x, y)$ in the xy –plane or moment m_{pq} such that the 2-D $(p + q)th$ –order moment of inertia is defined as:

$$m_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q g(x, y) dx dy \quad \forall p, q = 0, 1, 2, \dots, n \quad (4.8a)$$

where $g(x, y)$ is a piecewise continuous, bounded function *assumed to have nonzero values only in the finite part of the xy –plane*; therefore moments of all orders exist and the double moment sequence m_{pq} is uniquely determined by $g(x, y)$, and $g(x, y)$ is also uniquely determined by m_{pq} (36).

For a discrete, grayscale image with intensities $I(x, y)$, the ordinary moment (otherwise referred to as raw moment in some literatures) is defined as:

$$m_{pq} = \sum_x \sum_y x^p y^q I(x, y) \quad (4.8b)$$

It is a well-known fact that central moments are translational invariant due to the translational symmetry property of coordinates and normalized moments computed about the center of gravity make the distribution of the image mass at its origin. Hu (35) defines the continuous version of central moments and it is quoted,

$$\mu_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - \bar{x})^p (y - \bar{y})^q g(x, y) d(x - \bar{x}) d(y - \bar{y}) \quad (4.9)$$

Where \bar{x} and \bar{y} are the x – and y – coordinates of the image center of gravity given by:

$$\bar{x} = m_{10}/m_{00}, \text{ and } \bar{y} = m_{01}/m_{00} \quad (4.10)$$

For the discrete case, equation (4.9) can be re-written:

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q g(x, y) \quad (4.11)$$

Montgomery in (29) demonstrates the relationships between the raw moments and the central moments up to the first four orders and they are presented in (4.12):

$$\begin{aligned} \mu_{00} &= m_{00} = \mu, & \mu_{10} &= \mu_{01} = 0, & \mu_{20} &= m_{20} - \mu \bar{x}^2, \\ \mu_{11} &= m_{11} - \mu \bar{x} \bar{y}, & \mu_{02} &= m_{02} - \mu \bar{y}^2, & \mu_{30} &= m_{30} - 3m_{20} \bar{x} + 2\mu \bar{x}^3, \\ \mu_{21} &= m_{21} - m_{20} \bar{y} - 2m_{11} \bar{x} + 2\mu \bar{x}^2 \bar{y}, & \mu_{12} &= m_{12} - m_{02} \bar{x} - 2m_{11} \bar{y} + 2\mu \bar{x} \bar{y}^2, \\ \mu_{03} &= m_{03} - 3m_{02} \bar{y} + 2\mu \bar{y}^3. \end{aligned} \quad (4.12)$$

Where for the discrete case, $\mu_{00} = m_{00} = \mu$ (please see proof in DVD code function **invmom.m**)

Hu (35) presents the absolute orthogonal moment invariants which achieve orientation independence for pattern classification and further demonstrated that when they are merged with central moments, it is possible to derive a pattern classification that is independent of position, size and orientation.

These absolute orthogonal moment invariants are defined as

$$\delta_{pq} = \frac{\mu_{pq}}{\mu_{00}^\gamma} \quad (4.1.3)$$

where $\gamma = \frac{p+q}{2} + 1$ for the second and third order moments and for the second and third order moments, the six absolute orthogonal invariants are given as:

$$\begin{aligned} \sigma_1 &= \delta_{20} + \delta_{02}, & \sigma_2 &= (\delta_{20} - \delta_{02})^2 + 4\delta_{11}^2 \\ \sigma_3 &= (\delta_{30} - 3\delta_{12})^2 + (3\delta_{21} - \delta_{03})^2 & \sigma_4 &= (\delta_{21} + \delta_{12})^2 + (\delta_{21} - \delta_{03})^2 \\ \sigma_5 &= (\delta_{30} - 3\delta_{12})(\delta_{30} + 3\delta_{12})[(\delta_{30} + \delta_{12})^2 - 3(\delta_{21} + \delta_{03})^2] + \dots & (4.14a) \\ & \dots (3\delta_{21} - \delta_{03})(\delta_{21} + \delta_{03})[3(\delta_{30} + \delta_{12})^2 - (\delta_{21} + \delta_{03})^2], \\ \sigma_6 &= (\delta_{20} - \delta_{02})[(\delta_{30} + \delta_{12})^2 - (\delta_{21} + \delta_{03})^2] + 4\delta_{11}(\delta_{30} + \delta_{12})(\delta_{21} + \delta_{03}) \end{aligned}$$

The first three normalized moments in equation (4.14a) are used to further differentiate the landing target from background in the resulting marked image after corner segmentation (figure 4.4.b).

$$\sigma_1 = \delta_{20} + \delta_{02}, \quad \sigma_2 = (\delta_{20} - \delta_{02})^2 + 4\delta_{11}^2, \quad \sigma_3 = (\delta_{30} - 3\delta_{12})^2 + (3\delta_{21} - \delta_{03})^2 \quad (4.14b)$$

σ_1 is effectively the sum of the two second central moments with respect to the principal axes, often called the “spread” while σ_2 is the difference between the first two normalized central moments termed “slenderness”. Together with the other invariants, they are insensitive to translation, orientation and scale.

Montgomery in (12), (14) derives the eccentricity property in terms of the normalized moments and it is presented:

$$\varepsilon = \left[\frac{\mu_{02}\cos^2\alpha + \mu_{20}\sin^2\alpha - \mu_{11}\sin 2\alpha}{\mu_{02}\sin^2\alpha + \mu_{20}\cos^2\alpha - \mu_{11}\cos 2\alpha} \right]^2 \quad (4.15)$$

Where α is the object orientation defined as the angle between the x-axis and the principal axis of the object and can be mathematically evaluated as,

$$2\alpha = \tan^{-1} \left[\frac{2\mu_{11}}{\mu_{20} - \mu_{02}} \right] \quad (4.16)$$

5. Simulations and Experimental Results

Defining similar patterns as the ability to transform one pattern into the other without losing characteristic information, the simulation experiment was executed using three different approaches: first comparing the objects central moments (representing translational variation) and orientation moments (representing rotational error according to equation 4.16) between the UAV CSS corner marked image against actual positional patterns of detected corners (C_i, K_i) for $i = 1, 2, \dots, n$ in the offline calibrated image of figure 4.4b. Lastly, corner positional pairs in reference image are used to train a Restricted Boltzmann Machine by which the corner pairs of test images were tested and features reconstruction from the reference data was established.

To achieve best frame properties that suit the use of the developed program, the parallax serial terminal (37) was used to output servo commands to the CMUcam4 camera by turning on the *automatic pan* of the camera via the “AP 1” command, *automatic tilt* switched on with “AT 1”, its *automatic gain controller* was turned on via the “AG 1” command, *default brightness* was adjusted to 16 “CB 16” command because it was too dark initially. A list of the commands list of the cmucam4 is available in [(31)].




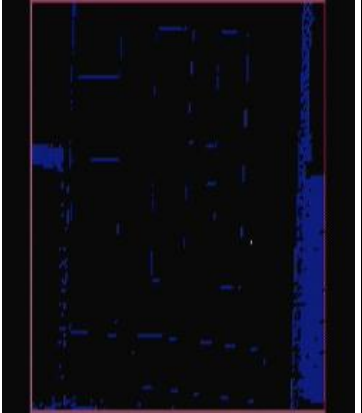

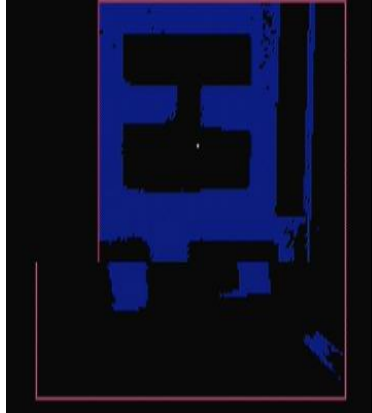
The automatic paparazzi program based on SPIN (see *appendix B*) was executed to dump bitmap (640×480 resolution) frames of the landing pad on the μ -SD card on-board the cam4 while the arducopter hovered above the landing target. These images were then collected and used in evaluating the performance of the algorithm.

The CMUcam4 operates by first adjusting to lighting conditions in the environment. This usually takes about 5 seconds from start-up. If light conditions in the room were to vary slightly, it automatically resets itself to

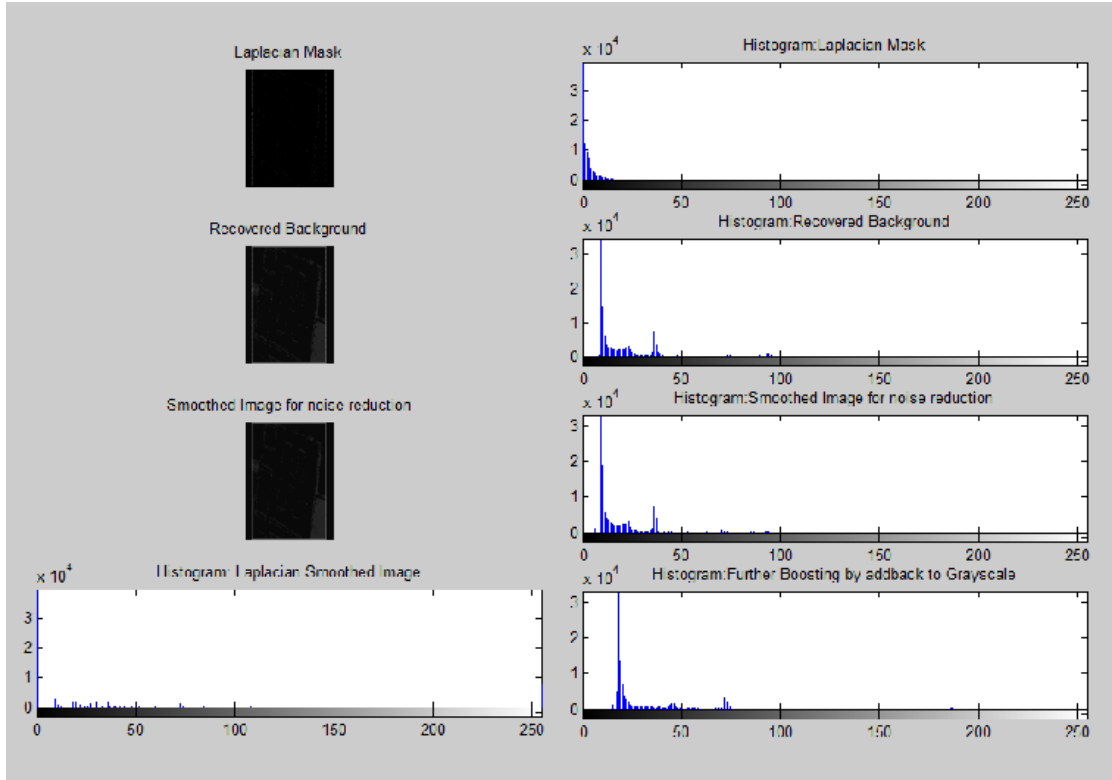
adjust to the new lighting condition. During the image acquisition process, even when auto gain and white balance were running only for a short period of time, it took the camera an average of 2 – 3 minutes to acquire the desired similitudes of the helipad image. Overall, landing pad images acquired were blurry, corner points were not abrupt and sharply defined as required by the curvature scale space algorithm making curves extraction extremely difficult. Thus image frames lacked rich features necessary for corner extraction. Thresholding the image was also an issue as the thresholding algorithm barely made distinction between target image and background after smoothing via the process discussed in section 4.1.1. At certain instances when corners got extracted, they were found to be separated from the actual landing target with corner position pairs were based on surrounding background noise.

Results from smoothing are displayed above and while applying the corner detection algorithm extracted certain features, they were based on falsely identified corners. Generally, for the features extraction process, the CMUcam4 makes a poor choice in accurate frame capture with the sort of features we need. An FPGA embedded frame grabber PXI8252/PCI-8252 from national instruments, NTSC/PAL Sony camera and wireless RF video modem could be used to extract the image in real-time during the landing approach of the UAV. This will ensure reliability in image acquisition for further processing. Due to time constraint and project deadline, these could not be integrated and implemented.

For the purpose of simulation, however, frames from landing target videos captured by the Sony Camera were extracted via the Matlab image acquisition toolbox for further processing.

		
<p>(a). Cmucam4 frame acquired with automatic gain turned off</p>	<p>(b) Cmucam4 frame with automatic pan turned on.</p>	<p>(c) Cmucam4 frame with automatic tilt turned on</p>
		
<p>(d) Image with Colour Tracking Feature Turned on</p>	<p>(e) Image Taken with Camera Contrast and Brightness adjusted during Colour Tracking</p>	<p>(f) Pan Parameters with proportional controller at 640 Derivative at 400 with Colour Tracking Off</p>

19 FIGURE 5.1 DIFFERENT CMUCAM4 FRAMES TAKEN WHILE ADJUSTING DIFFERENT PROPERTIES OF THE OMNIVISION CAMERA.



20. FIGURE 5.2 EXAMPLE SMOOTHENED IMAGE FROM THE CMUCAM4 FRAME

The resulting images were thresholded and fed through the CSS algorithm before corrupting the resulting corner marked images by additive white Gaussian noise⁴.

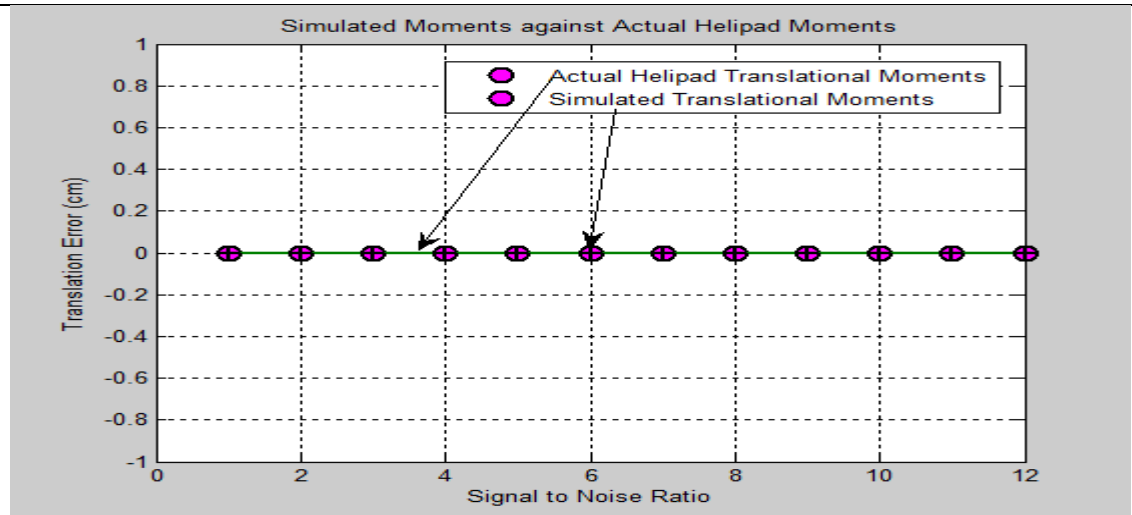
Invariant moments for the first, and second order moments of inertia for the resulting images were then generated along with the eccentricity, and orientation with respect to the landing target. Performance was evaluated against the calibrated offline image as a function of eccentricity error, euclidean distance between first two order moments and corner position patterns; and orientation and translation error.

5.1 Invariant Moments Sensitivity

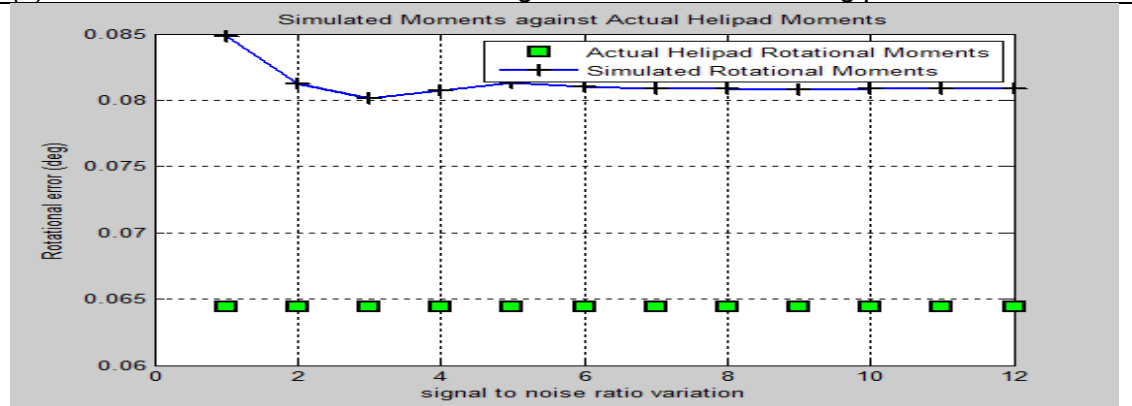
⁴ Averaging test images was explored but it was discovered they made for bad corner characterization therefore the course was abandoned.

First and second order normalized central moments for the test images was computed according to equations (4.14a) and since the central moments of inertia are insensitive to translation, we found a way to measure the degree of error along the translational axis of the simulated images.

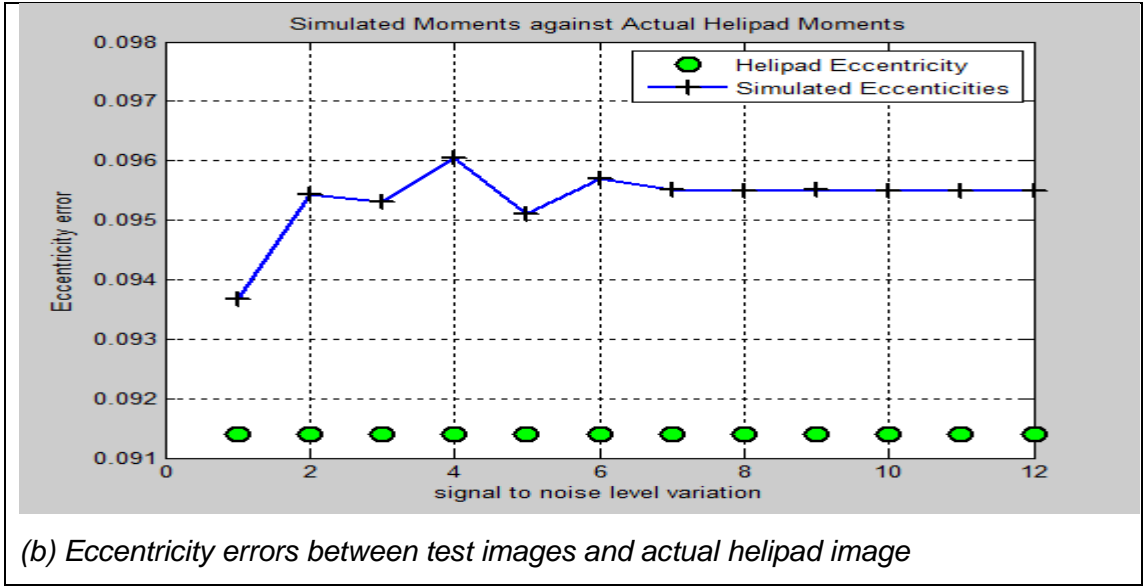
Based on equations (4.14) above, the following results were generated



(a) Translation Errors between test images and reference landing pad



(b) Rotational errors between test images and actual helipad image



21. FIGURE 5.3 EVALUATION OF INVARIANCE BETWEEN ACTUAL HELIPAD IMAGE AND CORRUPTED IMAGES

Normalized central moments take account of every pixel in the image to generate the corresponding moment of inertia use centroids about each axis of translation. One would therefore expect that with increasing levels of signal to noise ratio, the translational error term should stay relatively constant. This is validated in figure 5.7a.

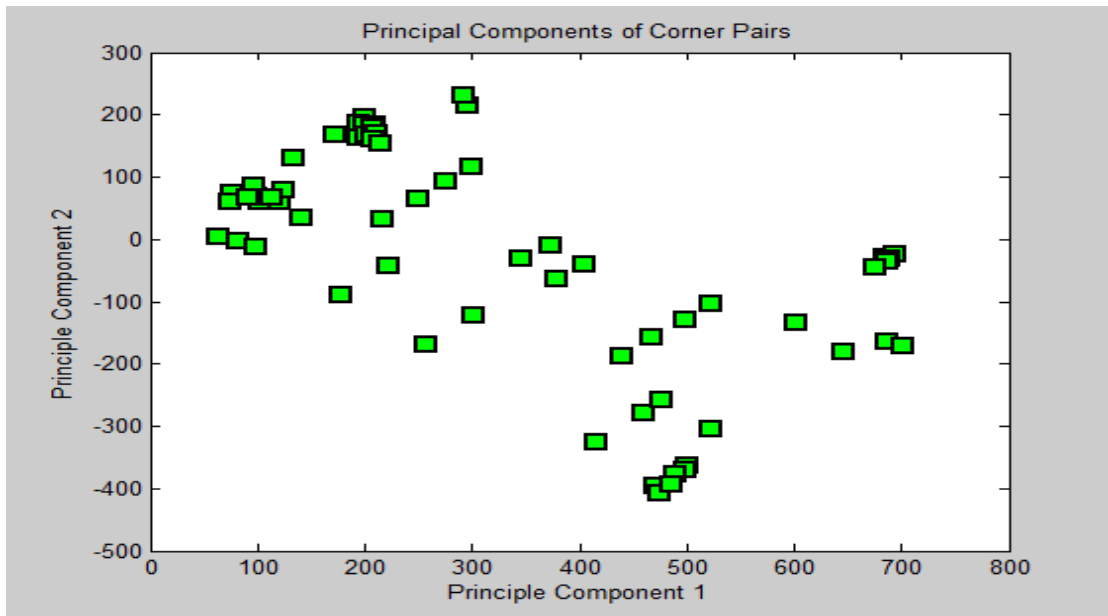
The object orientation is the angle between the principal axis of the target and the x-axis and defined though the minimization of the function:

$$R(\alpha) = \sum \sum_{(x,y) \in \mathcal{R}} [(y - \bar{x})\cos\alpha - (y - \bar{y})\sin\alpha]^2 \quad (5.1)$$

(x, y) being the image axes in the 2-D space \mathcal{R} . Subjective observation of figures 4.7 reveal that the invariant moments approach reveal close features matching between landing target and UAV camera. From the definition of eccentricity in equation (4.15) and object orientation in (4.16) we see why there is a higher sensitivity between the rotational and actual eccentricity estimates. (5.1) contains geometric properties of the image based on trigonometric operations and axial centroids. The minimization process in involves round-off errors in finding a suitable minima in the derivation of

angle α and pixel centroids is such that some features will be lost during computation. Similarly, the *eccentricity – estimates* are derived from second order and first order central moments which involve the *x and y* centroids, resulting squares and orientation headings.

To correct the errors along the rotational axis, improve the translation estimates and bolster corner points detection further, a neural network learning machine was proposed to force the machine to learn actual patterns.



22. FIGURE 5.4 PRINCIPAL COMPONENT ANALYSIS OF EXTRACTED CORNER PAIRS IN HELIPAD IMAGE

5.2 Artificial Neural Networks

Modelled after biological neurons, artificial neural networks (NNs) behave like functional biological neural systems using programming elements that segment the complexity in a given system through abstraction and identifying disjointed patterns thereby. The corner pair lists collected by the CSS algorithm early on is nonlinear, varied in the 2-dimensional space and cannot be easily modelled (see figure 4.9 for its principle components). Moreover, it has no definite input-output relationship and as a result, popular feed-forward neural networks such as the generalized linear model, or multilayer

perceptron networks are not convenient for modelling and pattern learning in this case.

Recurrent neural networks make a good case for the problem presented with the acquired data as they are highly applicable to tasks where internal memory is used to distinguish general inputs.

5.2.1 Boltzmann Machines

The Boltzmann uses the Boltzmann distribution in stochastic machines and statistical parallel search methods to derive a general learning rule for modifying the connection strengths used to *associate* knowledge about a task domain efficiently. It learns underlying constraints that are **UNIQUE** to patterns when it is shown an example of those patterns. By changing its connections based on example patterns shown, it is able to *create* an internal model which produces examples with the same probability distribution as that which it is shown. In Appendix D, the mechanisms of the Boltzmann machine are detailed

The Boltzmann machine presents a good theoretical computational model but does not fit into non-trivial problems as it takes time for the machine to gather equilibrium statistics which are exponential in growth with respect to machine dimensions. Also, connection strength becomes weak when elements being connected have their probabilities of activation between *off* and *on* states causing the strength of connections to be haphazard until activities get saturated (38).

By not allowing for intra-layer connections between hidden units (or testing data), Restricted Boltzmann Machines (or RBM's) overcome the challenges faced by the Boltzmann machine. Thus by training the RBM, activities of its

hidden units are treated as data for training a higher level RBM making it possible to train lots of hidden unit layers proficiently (39).

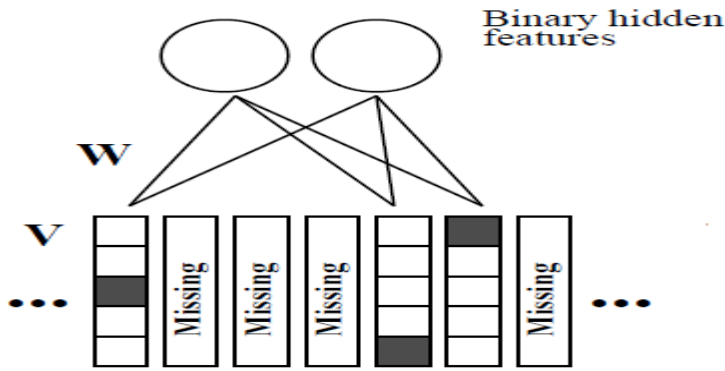
5.2.2 Restricted Boltzmann Machines

A restricted Boltzmann Machine is a class of two-layer undirected connections graphical model that allocates an energy state to each configuration of visible (V) and hidden (H) state vectors such that

$$E(V, H) = -B^T V - C^T H - V^T W H \quad (5.2)$$

where W is the matrix connection strength of weights of hidden and visible units representing the symmetric interaction parameter between features V and H ; B is the bias of the visible unit and C is the bias of the hidden unit (48).

It is an unsupervised efficient learning algorithm based on the deep belief network which learns connection weights used for training adjacent pairs of layers. Details about its working mechanism are listed in section 2 of appendix D.



[Adapted from^[53]]

23. FIGURE 5.5 INTERACTION OF VISIBLE AND HIDDEN UNITS VIA LEARNED WEIGHTS IN AN RBM

The visible units are used for training while the hidden layers are used for testing. Extracted corners were converted to binary via the *dmstandard* function and *dec2bin*® commands in Matlab (see attached DVD code) and

labeled dnG. To model each column of dnG, a conditional Bernoulli distribution was used to model input image features “ H ” as follows:

$$p(V_i^j = 1|H) = \frac{\exp(B_i^k + \sum_{j=1}^F H_j W_{ij}^k)}{\sum_{l=1}^K \exp(B_i^l + \sum_{j=1}^F H_j W_{ij}^l)} \quad (5.3)$$

where V is a $K \times m$ binary matrix for the i^{th} –visible units and j^{th} hidden units with $V_i^k = 1$ when a node is discovered as k and 0 otherwise. $H_j, \forall j = 1, \dots, F$ are the latent binary variables denoting the stochastic binary features different from image to image.

$$p(H_j = 1|V) = \sigma(B_j + \sum_{i=1}^m \sum_{k=1}^K v_i^k W_{ij}^k) \quad (5.4)$$

For training purposes, [53] defines a stochastic gradient ascent search in the log likelihood sense:

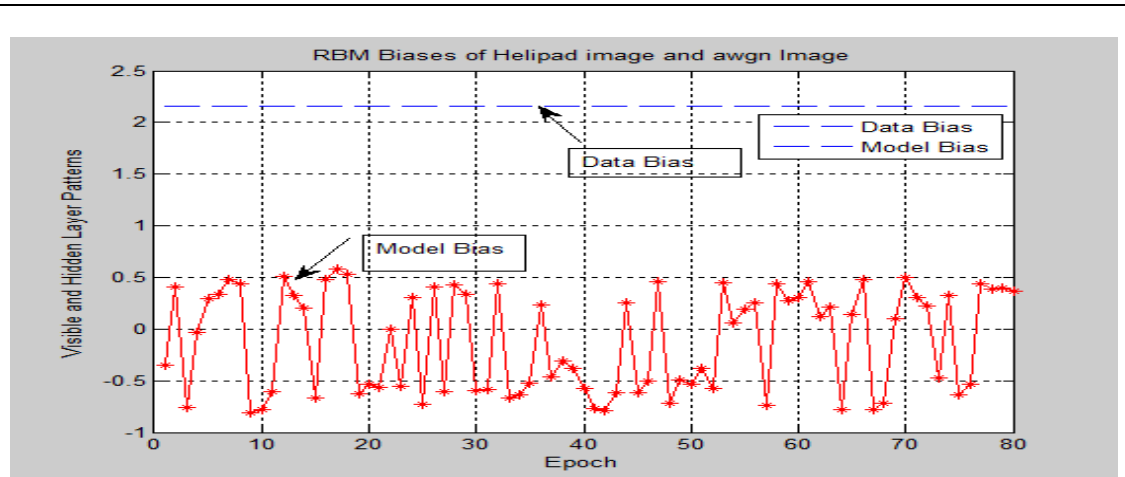
$$\Delta W_{ij}^k = \epsilon \frac{\partial \log(V)}{\partial W_{ij}^k} = \epsilon [\langle v_i^k H_j \rangle_{data} - \langle v_i^k H_j \rangle_{model}] \quad (5.5)$$

where ϵ is the learning rate and $\langle v_i^k H_j \rangle_{data}$ is the expectation that defines the frequency at which the feature j and binary rating k are on together when driven by input data; and $\langle v_i^k H_j \rangle_{model}$ is the expectation defined by the model. Log likelihoods are computed based on contrast divergence by Salakhutdinov (40)

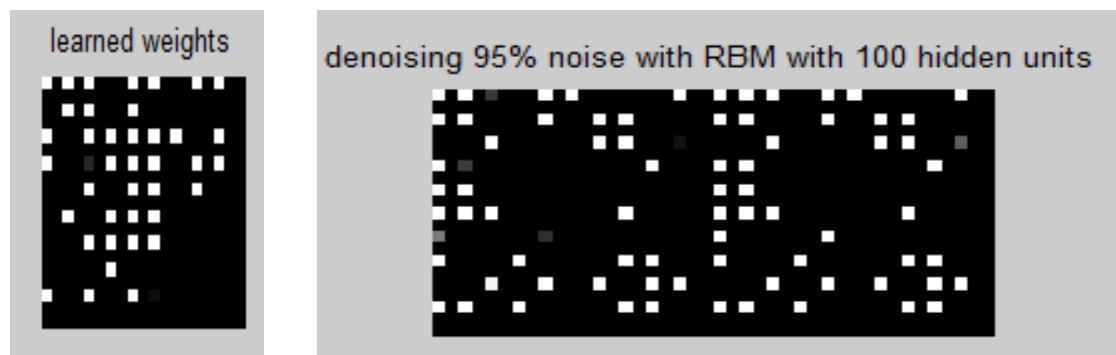
$$\Delta W_{ij}^k = \epsilon (\langle v_i^k H_j \rangle_{data} - \langle v_i^k H_j \rangle_T) \quad (5.16)$$

where $\langle . \rangle_T$ is the distribution of samples obtained by running the Gibb’s sampler in (5.18) initialized at the data for T full steps. (40), (41), (42), (43), (44) provide further details on restricted Boltzmann machines, deep belief networks and the persistent contrast divergence phenomenon which are used for pattern classification in this work.

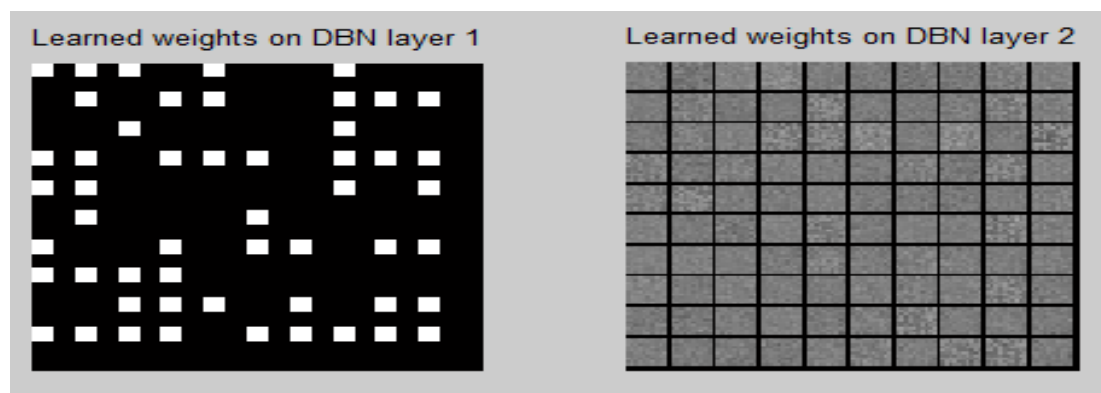
To support the pattern recognition challenge using RBM's, a code provided by Andrej Karpathy (45) was rewritten to suit the problem context and the reconstruction results are presented in *figure 5.7*.



(a) RBM Sensitivity to helipad corner pairs as seen by UAV (test data) and reference



Left(b) learned corner weights based on RBM. Right (c) result of denoising image with 100 testing data based on Bernoulli RBM



Left (d) Learned weights based on first layer of deep belief network; Right (e) Learned weights

24. FIGURE 5.7 PATTERN CLASSIFICATION VISUALIZATIONS

Based on the principal components analysis result of figure 5.4, we see why corner positions are irregular as displayed in figure 5.7. Moreover, actual corner positions of the target have variants of false corners albeit small, and these are also fed into the Bernoulli based RBM. In figure 5.7(d) and (e) we use a deep belief network with two 100 hidden layers. (d) shows the learned weights at the end of our experiment for the first layer while (e) shows result for the second layer. Although classification error was the same, subjective observation of the learned weights for the first layer suggest patterns are better reconstructed compared to the RBM results. On the overall, classification error for supervised learning between hidden layer (test image corner positions) and visible layers (established corner points) was found to be 36.7%.

6. Conclusions and Recommendations

The aim of this project was to demonstrate the autonomous landing of an unmanned aerial vehicle using camera vision. In this thesis, the demonstration of the take-off, stabilization and flight to the predetermined landing pad was presented. While the UAV was positioned above the landing pad using expert knowledge, the camera sensor board system acquired images of the target.

6.1 Accomplishments

In the course of implementing the segmentation algorithm, we found the cmucam4 sensor acquired images were inconvenient to fit into the CSS algorithm. Due to project deadline exigencies, physical testbed implementation of landing target image processing could not be performed as orders for adaptable camera and frame grabbers take long before arrival.

However, offline test images were acquired and corrupted by additive white Gaussian noise based on different signal to noise ratios. For both the reference image and corrupted images, we first subsampled image size to make for computational efficiency and based on the default CMUcam4 output resolution, we found that 640×480 pixels were ideal. The subsampled images were converted to grayscale images. Histogram equalization, while it did even out the contrast in images, was redundant at higher levels of the segmentation process making corners and other interesting features vanishing. To overcome this, the Laplacian filter was used first to extract respective image edges; second, the result was then added back to the original image for full image recovery before a 3×3 averaging filter was used to smoothen the gradient of the Laplacian image. To get the best features in

both the smoothened and Laplacian differentiated image, we added the two images together which was then globally thresholded at 85%.

The scheme described ensured features such as edges and corners were preserved under transformations in a cooperative environment⁵. Although the algorithm was much too complex for the CMUcam4 images, we found that at certain levels of smoothing, certain corner points were extractable even though they existed outside the scope of the landing target and were mostly background noise acquired in the process of the sensor adjusting for frame capture of the landing target.

Using the Gaussian noised test images the thresholded image was fed through the curvature scale space algorithm extracting corner position pairs and corner marked image outputs in the process. We generated invariant moments based on various noised images and we compare eccentricity, translation and rotation errors to those of the reference image which we had calibrated early on. The results show distinct properties were preserved under Euclidean motion with minor sensitivities in orientation and eccentricity.

To safeguard pattern reconstruction in the event of high calibration errors between the test and reference images, we adopted a neural network pattern reconstruction algorithm using the restricted Boltzmann machine. Based on various test images, we found the classification error to be on the average of 36% by the Boltzmann machine.

One limitation experienced in feeding the Boltzmann machine with actual image data is in the scaling of helipad image dimensions and binary conversion such that the RBM algorithm can recognize the image as a visible layer. Furthermore, other deep belief network machines could be tested and

⁵ Cooperative environments are those where transformations and natural disturbance do not change trackable image features such as sites with stationary objects, marked images that the vehicle may use during the landing task.

evaluated for pattern recognition in corner positions of UAV frames with respect to the landing target.

6.2 Project Milestones and Recommendations

Within the broader aim of developing a visual sensing machine in the autonomous landing of a RUAV, this project has developed a flying quadrotor and low-level image processing mechanism for the vision subsystem of the UAV.

The quadrotor's first RC-based flight was demonstrated on July 10, 2012. By tuning the on-board PID controller, the initial yaw and roll gyrations in mid-air were stabilized in the week that followed. Due to the delays in product orders and technical challenges faced in the course of the project, the CMUcam4 was only coupled twelfth week of the project. The magnanimity of the initial project aims and objectives were realised during this time and focus was laid upon developing telemetry-based command-control of the quadrotor system.

Using Xbee pro modules, flight and navigation to predetermined landing pad location was demonstrated in week 13 of the project but then the receiver module of the telemetry system crashed shortly afterwards prompting the order of the Xbee series 2 modules for replacement. Sustained hover above landing target is possible if the javascript included in appendix B is executed in the mission planner. It has been executed using a USB connection to the autopilot for command and control but the UAV was tethered to the ground as the reach of the cable was limited.

The cmucam4 is a good tool for trivial data and video capture of ground target observation but its image processing capacity is low for the task objectives presented. Camera calibration could be explored to enhance the accuracy of features detection by the cmucam4 but due to time-imposed constraint, this area could not be explored. Instead, we assumed additive white Gaussian

noise of simulated helipad images as UAV camera frames in this project and based on this segmentation and invariant moments analysis are performed.

By attempting to derive complete, actual helipad images based on example patterns fed into the RBM machine, classification errors in patterns matching was found to be $35 \pm 1\%$. This is a useful approach in actual implementation but the accuracy could be further improved if corner pair positions are correctly labelled instead of the random pair labelling we assumed.

Therefore the two accomplishments presented in this work are

- improve existing UAV for autonomous navigation purposes; navigate to a predetermined landing target, hover above landing target during which the UAV camera acquires videos and camera frames;
- apply invariant features and corner segmentation on UAV camera frames and compare results with calibrated helipad image; Learn actual helipad features if variance in matching is high;

Results presented show features in an object remain constant from image to image. In a rough terrain such as landing in on a rooftop or rough moving landing target this might not be true.

The monocular tracking presented is meant to update its matching template after each camera frame. In the event of camera line of sight blockade the updated template could lose information and be incorrectly updated. This can eventually lead to wrong pose estimation and motion which might damage the UAV during landing.

To correct this, a redundant system of cameras could be used so that the vision system would be less sensitive to incorrectly tracked features.

We have also assumed local knowledge of the landing area. In dynamic UAV flights, this is not a privilege meaning mission planning, obstacles recognition

and avoidance, simultaneous localization and mapping have to be considered in the robot's navigation.

6.3 Future Work

Using stereo cameras, estimation errors will depend on camera motions with respect to the observed plane. Features matching and image velocities from frame to frame of the camera can be used to extract UAV motion and structure. Based on these, a controller input can be designed in the landing process. We present the differential case of the ego motion problem (10), (46), (47) and relate how it could be integrated into this work in the future.

6.3.1 Geometric Visualization

For accurate robot pose accurate estimation the optical axis of the camera is assumed to be identical to the z-axis of the UAV frame. Denoting the aerial vehicle frame as Ω and the ground frame as Θ , and assuming the coordinates of a point $P = [x, y, z]^T$ in the inertial frame is marked by a prime as in $x' \in \mathbb{R}^3$ and a point in the camera frame is the same alphabet but without a prime: $x \in \mathbb{R}^3$, a mapping from a $[\cdot]_{\times}$ (3D vector \mathbb{R}^3) to a 3×3 matrix is

$$[x_1, x_2, x_3]^T = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}. \quad (6.1)$$

From cross product of vectors, for two vectors $x, y \in \mathbb{R}^3$, we find $x \times y = [x]_{\times} y$.

Assume a special Euclidean group $SE(3) = \{(T, R) | T \in \mathbb{R}^3, R \in SO(3)\}$ describes the UAV's motion on a smooth curve with $SO(3) \in \mathbb{R}^{3 \times 3}$ as the special orthogonal group of rotational matrices with +1 determinant; and let T , and R represent the position and orientation of the camera with respect to the inertial frame at time t , the spatial points in the inertial frame and coordinates in the camera frame are related by

$$x' = Rx + T \quad (6.2)$$

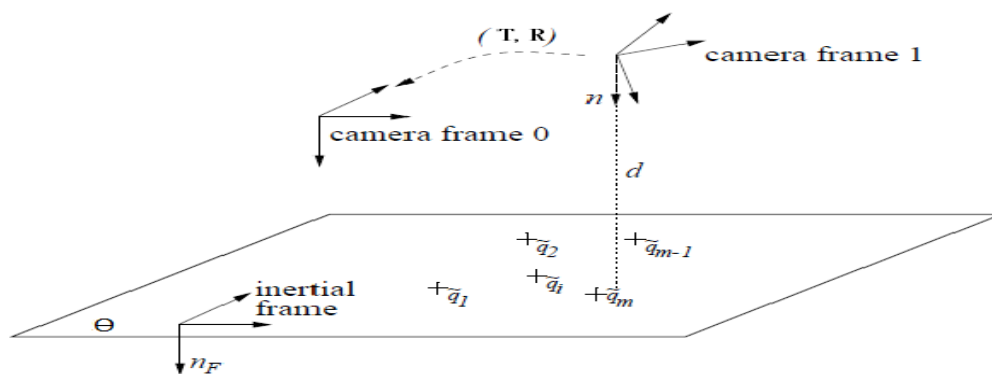
where $\mathbf{x} = (x \ y \ z)^T$ and $\mathbf{x}' = (x' \ y' \ z')^T$

By differentiating (6.2), the linear and angular velocities of the camera frame relative to the inertial frame defined by $v, w \in \mathbb{R}^3$ are found to be related by:

$$\dot{x} = -\hat{w}x - v \quad (6.3)$$

$$\text{where } \hat{w} = R^T \dot{R}, \text{ and } v = R^T \dot{T} \quad (6.4)$$

The perspective camera projection model is assumed because depth differences in the visual scene are taken into account as opposed to the



25. FIGURE 6.1 GEOMETRY OF CAMERA FRAME WITH RESPECT TO LANDING PLANE [CREDIT: 12]

orthographic model thus ensuring corner features are preserved.

Assumed a distance of unity away from the camera's optical centre of image feature the perspective projection onto this surface is given by the mapping π_θ from \mathbb{R}^3 to the **projective plane** $\mathbb{R}P^2$ that is,

$$\pi_\theta: \mathbb{R}^3 \setminus \{0\} \rightarrow \mathbb{R}\theta^2$$

$$(x \quad y \quad z)^T \rightarrow \left(\frac{x}{z}, \quad \frac{y}{z}, \quad 1\right)^T \quad (6.5)$$

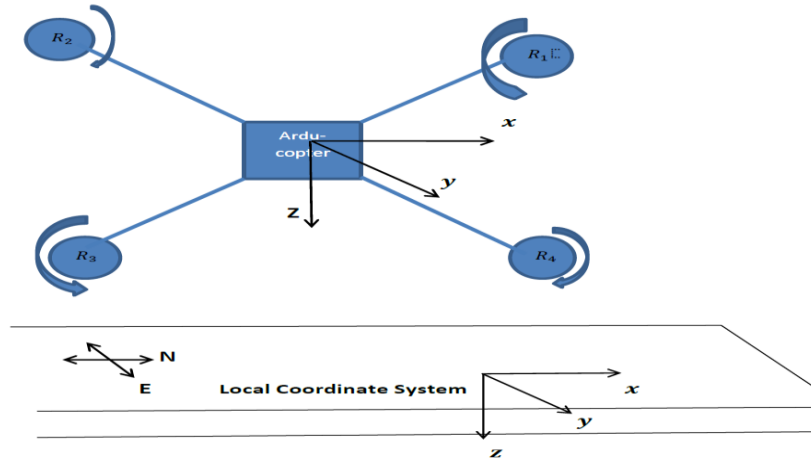
Therefore, the image of a point x taken by the camera is then

$$q = \pi(x) \quad (6.6)$$

If the optical axis is denoted by $e_3 = (0 \ 0 \ 1)^T$, then (5.2.3) can be rewritten as: $(I - qe_3^T)x = 0$ (6.7)

6.3.2 Framework of Coordinate Systems

Taking the local ground coordinate system as the reference as in figure 5.9, it is found that the inertial ground frame is aligned with the earth's magnetic coordinate system. The ground is said to be local because its origin is at a



26. FIG 6.2 QUADROTOR INERTIAL AND LOCAL COORDINATE SYSTEMS

bounded region where the geometrical plane is level; and choose the UAV's origin at its centre of mass with attitude angles pointing towards the positive coordinate system.

R in (6.2) is the standard 3×3 Euler matrix (48) and is a function of three Euler angles (α, β, γ) used to rotate a point from the UAV frame to the inertial coordinate frame defined as

$$\Delta(\alpha, \beta, \gamma) = \begin{bmatrix} c\beta c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma & c\alpha s\beta c\gamma + s\alpha \gamma \\ c\beta s\gamma & s\alpha s\beta + c\alpha c\gamma & c\alpha s\beta s\gamma - s\alpha c\gamma \\ -s\beta & s\alpha c\beta & c\alpha c\beta \end{bmatrix} \quad (6.9)$$

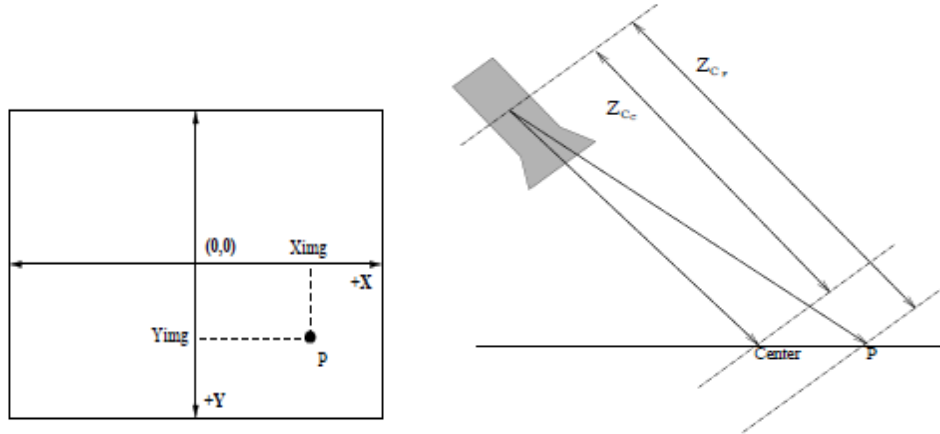
With the notation $s\alpha = \sin(\alpha)$ and $c\alpha = \cos(\alpha)$ and T_Q^L is the equivalent UAV coordinate in the ground frame.

6.33 Camera Setup and Coordinate Transformations

By fixing the UAV camera directly under the UAV with the camera lens pointing downwards, pitch offset between the UAV and camera coordinate frames can be eliminated. Assuming the origin of the camera coordinate frame at the focal length of the camera UAV position can be calculated about the camera's focal point and feature points on the landing pad to be tracked are assumed to be at equal height above the ground as the camera establishing UAV attitude and altitude above ground target can be estimated based on camera measurements.

Suppose R_C^Q represents the rotation matrix and T_C^Q denote the translation matrix of the camera frame around the UAV axes, a point P_C in the camera frame could be translated to the arducopter frame by the following relation:

$$P_Q = R_C^Q P_C + T_C^H \quad (6.10)$$



27. FIGURE 6.3- (A) CAMERA IMAGE COORDINATE SYSTEM (B) FEATURE POINTS IN ACQUIRED IMAGE

Thus, a point $P_I = (P_{I,x} \ P_{I,y} \ P_{I,z})^T$ in the camera image frame can be transformed to a point $P_C = (P_{C,x} \ P_{C,y} \ P_{C,z})^T$ in the camera frame by (11):

$$P_C = P_{I,z} \begin{bmatrix} \frac{P_{I,x}}{F} \\ \frac{P_{I,y}}{F} \\ 1 \end{bmatrix} \text{ or } \begin{bmatrix} P_{C,x} \\ P_{C,y} \\ P_{C,z} \end{bmatrix} = P_{I,z} \begin{bmatrix} \frac{P_{I,x}}{F} \\ \frac{P_{I,y}}{F} \\ 1 \end{bmatrix} \quad (6.11)$$

where F is the focal length of the camera defined as:

$$F = \frac{L}{\tan^{-1}\left(\frac{\theta}{2}\right)} \quad (6.12)$$

L is the width of the image plane and θ the camera's field of view

6.3.4 Algorithm Formulation

The *ego-motion estimation problem* involves tracking features in the landing pad by locking onto it via the camera sensing vision. In the *differential ego motion estimation algorithm* the velocities of image points in a long sequence of images are used to estimate the rotational and translational velocities of the camera. The *essential matrix* is minimized and motion and structure parameters are recovered through single value decomposition of eigenvalues.

6.3.5 Differential Ego Motion Estimation

Using the estimates of velocities, the calculation of control inputs for the UAV is possible. If the camera is subjected to a rigid motion with linear and angular velocities $v(t)$, and $\omega(t)$, the coordinates of coplanar points $\{x'_i\}_{i=1}^k$ in the instantaneous frame of the camera is defined by:

$$\dot{x}_i(t) = -\left[\hat{\omega} + \frac{1}{d} v n_{\theta}^T\right] x_i(t), \quad i = 1, \dots, k \quad (6.13)$$

The inner matrix in (6.13) is called the *differential planar essential matrix*, i.e.,

$$B = \left[\hat{\omega} + \frac{1}{d} v n_{\theta}^T\right] \quad (6.14)$$

because it has all motion and structure parameters necessary for estimating the pose of the UAV robot and satisfies the constraint:

$$\dot{q}_i = -(I - q_i e_3^T) B q_i, \quad i = 1, \dots, m \quad (6.15)$$

$\{q_i(t), \dot{q}_i(t)\}_{i=1}^m$ being the image points and fixed point velocities in the plane of concern.

Denoting the matrix of linear elements as in (6.15) by G , it follows that the $rank(G) = 8$ if and only if $\{x'_i\}_{i=1}^k$ are in general configuration [(46)]. Matrix B can only be uniquely determined from measurements of the image if and only if the points $\{x'_i\}_{i=1}^k$ are in general configuration [17].

Thus by estimating parameters, w and $v, \{\widehat{\omega}, \frac{v}{d}, n_\theta\}$, motion parameters can be recovered up to at most 2 physically possible solutions given the matrix $B \in \mathbb{R}^{3 \times 3}$.

6.4 Concluding Remarks

A cheap, passive and information-rich computer vision sensor is gaining importance and research interests where it can be embedded in a feedback loop for control purposes by the day. Vision-based aerial vehicles provide remote, pilotless visual information which when combined with specific movements can generate topographic overview useful in policing operations, assets monitoring, and media production.

This dissertation has presented the promising results of invariant feature moments generation based on the curvature scale space corner detection on a landing pad for the autonomous landing of a vision-based UAV robot.

Due to project deadlines constraints, the ego motion estimates useful for controller input design for autonomous landing of the UAV could not be developed. Background for this in future work has been provided to make design and implementation easy.

References

1. Unmanned Aerial Vehicles. U.S. Department of Defense. . [Online]
<http://www.defense.gov/specials/uav2002/uavpage01.html..>
2. **Preston, Darrell.** Drones Take to American Skies on Police, Search Missions. [Online] [Cited: 2 June 2012.] <<http://www.businessweek.com/news/2012-05-31/drones-take-to-american-skies-on-police-search-missions>> .
3. The Case for Using UAVs for your Oil Security. [Online] [Cited: 22 June 2012.] <<http://www.hawkaerospace.eu/Default.aspx?tabid=239&articleType=ArticleView&articleId=42>>.
4. Making UAV and aerial robotics easy and affordable. [Online] [Cited: 13 August 2012.] <http://jdrdrones.com>.
5. **Schmidt, Michael David.** *Simulation and Control of a Quadrotor Unmanned Aerial Vehicle*. Kentucky, USA : University of Kentucky, 2011.
6. **Jay Farrell, Matthew Barth, Randy Galijan, Jim Sinko.** *GPS/INS Based Lateral and Longitudinal Control Demonstration: Final Report*. s.l. : California PATH Research Report.
7. **Sylvester, David Tom.** *Development of a Vision Based Landing System for Use on a Rotary Wing Unmanned Aerial Vehicle*. Sheffield, UK : Department of Automatic Control & Systems Engineering, The University of Sheffield, 2012.
8. **Mbaekube, Ugonna.** *Altitude Control and Stabilization of a Quadrotor*. Sheffield, UK : Department of Automatic Control & Systems Engineering, The University of Sheffield, 2012.

9. *Vision Guided Landing of an Unmanned Air Vehicle*. **Omid Shakernia, Yi Ma, T. John Koo, Joao Hespanha, S. Shankar Sastry**. Phoenix, Arizona : Proceedings of the 38th Conference on Design & Control, 1999.
10. **Hintze, Joshua**. *Autonomous Landing of a Rotary Unmanned Aerial Vehicle in a Non-Cooperative Environment Using Machine Vision*. Department of Electrical and Computer Engineering, Brigham Young University. 2004. Thesis.
11. *Vision Based Terrain Recovery for Landing Unmanned Aerial Vehicles*. **Marci Meingast, Christopher Geyer, Shankar Sastry**. Atlantis, Paradise Island, Bahamas : 43rd IEEE Conference on Decision and Control, 2004.
12. *Vision-based Autonomous Landing of an Autonomous Aerial vehicle*. **S. Saripalli, J.F. Montgomery and Gaurav S. Sukhatme**. Washington D.C. : in Proceedings on Robotics & Automation of the 2002 IEEE International Conference, May 2002.
13. **Vilas K. Chitrakaran, Darren M. Dawson, Jian Chen, and Matthew Feemster Member, IEEE**. Vision Assisted Autonomous Landing of an Unmanned Aerial Vehicle. *IEEE explore*. [Online] 12-15 December 2005. [Cited: 01 August 2012.] <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1582365>.
14. **S. Saripalli, J.F. Montgomery, G.S. Sukhatme**. Visually Guided Landing of an Unmanned Aerial vehicle. *IEEE transactions on Robotics and Automation*. June 2003, Vol. 19, Vol. 3.
15. **E.D. Dickmanns, F.R. Schell**. *Autonomous Landing of Airplanes by Dynamic Machine Vision*. Munchen, Germany : Universitat der Bundeswehr.
16. **Omead Amidi, Takeo Kanade, Keisuke Fujita**. A Vision Odometer for Autonomous Helicopter Flight. *Robotics and Autonomous Systems* 28. 1999.
17. **Amidi, Omead**. *An Autonomous Vision-Guided Helicopter*. Pittsburg : PhD Thesis Submitted to the Department of Electrical and Computer Engineering, Carnegie Mellon University, 1996.

18. **T.J. Koo, F. Hoffmann, H. Shim, B. Sinopoli and S. Sastry.** *Hybrid Control of an Autonomous Helicopter.* Berkeley, USA : Department of Electrical Engineering and Computer Sciences, University of California.

19. **Omid Shakernia, Yi Ma, T. John Koo, Shankar Sastry.** *Landing an Unmanned Air Vehicle: Vision Based Motion Estimation and Nonlinear Control.* s.l. : Asian Journal of Control, 1999. Vol. 1, No. 3, pp. 128 - 145.

20. **C. Sharp, O. Shakernia, and S. Sastry.** *A Vision System for Landing an Unmanned Aerial Vehicle.* Seoul, South Korea : Proceedings of the 2001 IEEE , 2001.

21. Historic UC Berkeley Robotics and Intelligent Machines Lab Home Page (1993-2008). . [Online] [Cited: 04 August 2012.] <http://robotics.eecs.berkeley.edu/bear..>

22. *A Theory of Multiscale, Curvature-Based Shape Representation for Planar Curves.* **F. Mokhtarian, and F. Mohanna.** 8, s.l. : IEEE transactions on pattern analysis and machine intelligence, 1992, Vol. 14.

23. *Curvature Scale Space Corner Detector with Adaptive Threshold and Dynamic Region of Support.* **Yung, X.C. He and N.H.C.** Hong Kong : Department of Electrical and Electronic Engineering.

24. *Corner detector based on global and local curvature properties.* **Yung, Xiao Chen He and Nelson H.C.** 5, Hong Kong : Optical Engineering, May 2008, Vol. 47.

25. **Rosenfield, L. Kitchen and A.** Gray Level corner detection. *Pattern Recognition Letters*, pp. 95 - 102. 1982.

26. *Determination of ego-motion from matched points.* **Harris, C.** Cambridge, UK : : In Proceedings 3rd Alvey Vision Conference, 1987.

27. *Determination of ego-motion from matched points.* **Harris, C.** Cambridge, UK : : In Proceedings 3rd Alvey Vision Conference, 1987.

28. *A new approach to low-level image processing*. **Brady, S. Smith and J. 1**, s.l. : International Journal of Computer Vision, 1997, Vol. 23.
29. *Vision-Based Autonomous Landing of an Unmanned Aerial Vehicle*. **S. Saripalli, J.F. Montgomery, G. Sukhatme**. Washington D.C. : Proceedings of the 2002 IEEE International Conference on Robotics and Automation, May 2002.
30. [Online] DIYdrones. <http://code.google.com/p/arducopter/wiki/ArduCopter>.
31. **Kwabena Agyeman, Anthonu Rowe**. Overview - CMUcam4 - CMUcam: Open Source Programmable Embedded Vision Sensors. [Online] <http://cmucam.org/projects/cmucam4>.
32. [Online] RevoElectrix. http://www.revoelectrix.com/lipo_terms.htm.
33. **Rafael C. Gonzalez, Richard E. Woods**. *Digital Image Processing*. Upper Saddle River, New Jersey : Prentice Hall Inc., 2002. 0-201-18075-8.
34. R2012a Documentation --> Image Processing Toolbox. *Mathworks*. [Online] <http://www.mathworks.co.uk/help/toolbox/images/ref/regionprops.html#bqkf8hf>.
35. **Hu, Ming-Kuei**. Visual Pattern Recognition by Moment Invariants. *IRE Transactions on Information Technology*. 1962, Vols. IT-8, PP. 179 - 187.
36. **Papoulis, A.** *Probability, Random Variables, and Stochastic Processes*. New York : McGraw-Hill, 1965.
37. Propeller Downloads. [Online] <http://www.parallax.com/tabid/832/Default.aspx#Software>.
38. Simulated Annealing. *Wikipedia*. [Online] [Cited: 27 August` 2012.] http://en.wikipedia.org/wiki/Simulated_annealing.
39. *A Learning Algorithm for Boltzmann Machines*. **David H. Ackley, Geoffrey E. Hinton and Terrence J. Sejnowski**. 9, s.l. : Cognitive Science 9, 1985.

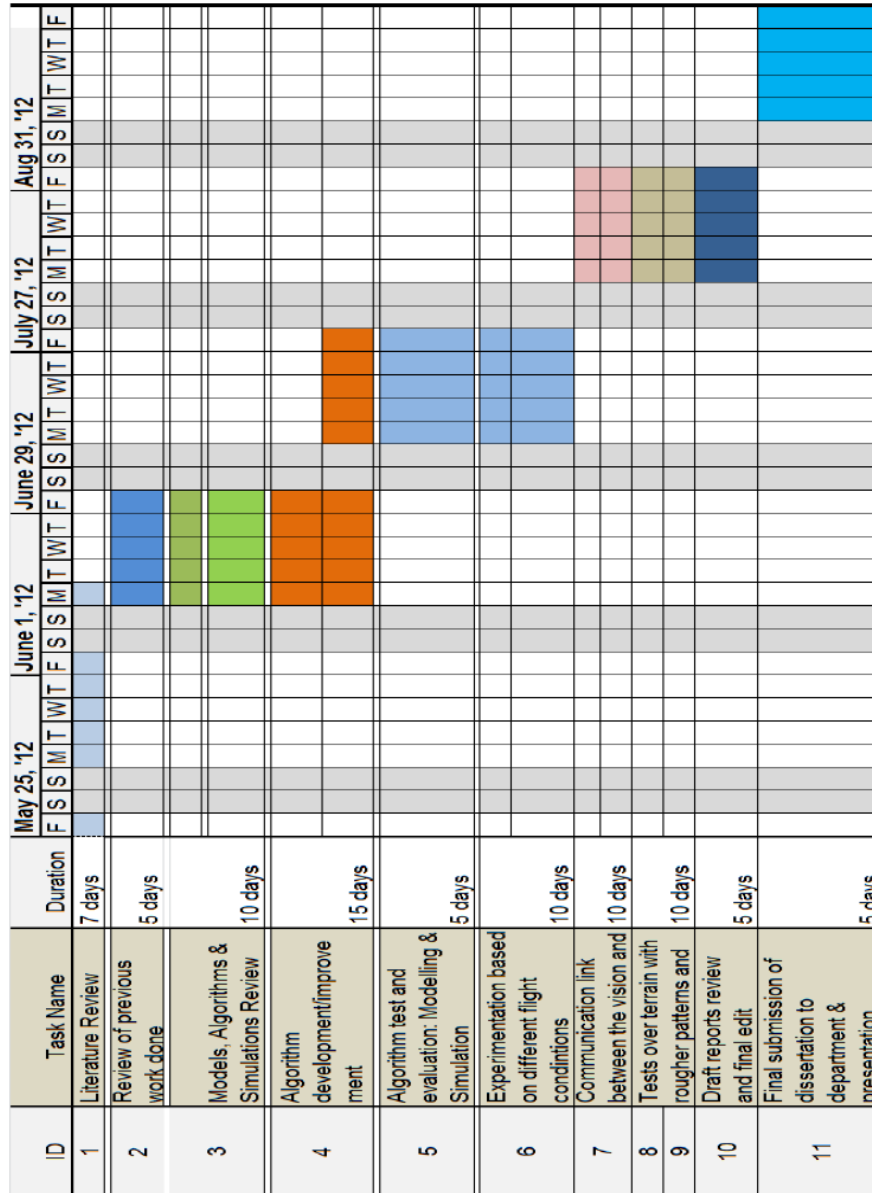
40. **R. Salakhutdinov, A. Mnih, G. Hinton.** *Restricted Boltzmann Machines for Collaborative Filtering*. s.l. : University of Toronto, Ontario M5S 3G4 Canada.
41. *Context-Dependent Pre-trained Deep Neural Networks for Large Vocabulary Speech Recognition*. **George E. Dahl, Dong Yu, Li Deng and Alex Acero.** s.l. : IEEE Transactions on Audio, Speech and Language Processing.
42. *Learning deep architectures for AI*. **Bengio, Y.** 1, s.l. : Foundations and Trends in Machine Learning, 2009, Vol. 2.
43. *Information Processing in Dynamical Systems: Foundations of harmonic Theory*. **Smolensky, P.** pp. 194 - 281, s.l. : Parallel Distributed Processing, 1986, Vol. 1.
44. **Tieleman, Tijmen.** *Training Restricted Boltzmann Machines using Approximations to the Likelihood Gradient*. Toronto : Department of Computer Science, University of Toronto, Toronto, Ontario M5S 3G4 Canada.
45. Code for training restricted Boltzmann machines (RBM) and Deep Belief Networks. *Google Projects*. [Online] [Cited: 28 August 2012.]
<http://code.google.com/p/matrbm/downloads/list>.
46. **J. Weng, T.S. Huang, N. Ahuja.** *Motion and Structure from Image Sequences*. s.l. : Springer-Verlag Berlin, 1993. ISBN 3-540-55672-9.
47. *Motion and structure from motion from point and line matches*. **O.D. Fauregas, F. Lustman and G. Toscani.** s.l. : IEEE International Conference on Computer Vision, 1987.
48. **Roskam, Jan.** *Airplane Flight Dynamics and Automatic Controls*. s.l. : Design, Analysis and Research Corporation, 3rd edition, 2001.
49. **Hopfield, John J.** Hopfield Networks. *Scholarpedia*. [Online] Princeton University, NJ, USA, 2007. [Cited: 27 August 2012.]
http://www.scholarpedia.org/article/Hopfield_network.

50. **Scheve, Tom.** *How the MQ-9 Reaper Works.* [Online] [Cited: 2 June 2012.]
<<http://www.howstuffworks.com/reaper1.htm>> .
51. Small, Unmanned Aircraft Search for Survivors in Katrina Wreckage, Press Release 05-160 . [Online] [Cited: 22 June 2012.]
<http://www.nsf.gov/news/news_summ.jsp?cntn_id=104453&org=OLPA&from=news>.
52. Pipeline Monitoring & Oil and Gas Security. [Online] [Cited: 22 June 2012.]
<<http://www.aeronautics-sys.com/?CategoryID=259&ArticleID=188>> .
53. RQ-4 Euro Hawk UAV Ready for Takeoff. [Online] 06 July 2012. <
<http://www.defenseindustrydaily.com/euro-hawk-program-cleared-for-takeoff-03051/>>.
54. *Landing on a Moving Target using an Autonomous Helicopter.* **Sukhatme, Srikanth Saripalli and Gaurav.** s.l. : in Proceedings of the International Conference on Field and Service Robotics, July 2003.
55. Historic UC Berkeley Robotics and Intelligent Machines Lab Home Page (1993-2008). [Online] [Cited: 04 Aug 2012.] <http://robotics.eecs.berkeley.edu/bear>.
56. *Linear Differential Algorithm for Motion Recovery: A Geometric Approach.* **Yi Ma, Jana Kosecka and Shankar Sastry.** 36(1), 71-89, Berkeley, CA : International Journal of Computer Vision; Kluwer Academic Publishers, 2000.
57. Unmanned Aerial Vehicles. [Online] U.S. Department of Defense. [Cited: 02 July 2012.] <http://www.defense.gov/specials/uav2002/uavpage01.html>.
58. Mathworks: Document > Image Processing Toolbox > User's Guide > Resizing an Image. [Online] [Cited:] <http://www.mathworks.co.uk/help/toolbox/images/f12-12267.html>.

59. **Ramesh Jain, Rangachar Kasturi, Brian G. Schunck.** *Machine Vision*. Boston, Massachusetts : McGraw-Hill, 1995.

Appendix A – Initial Project Gantt Chart

GANTT CHART SHEDULE OF PROJECT WORK



A.1 TASKS IDENTIFICATION AND OBJECTIVES

TASK NAME	TASK DESCRIPTION	OBJECTIVES AND TIMELINE
1. Background Research of Unmanned Aerial Systems <ul style="list-style-type: none"> •Explore Unmanned Ground Vehicles •Explore Unmanned Aerial Vehicles 	<ul style="list-style-type: none"> •To acquire a broad idea of the many paths the research work could follow; •To give student a fair idea of current limitations in different fields of research in unmanned aerial systems 	<ul style="list-style-type: none"> •Possible Project Aims and Objectives [First Week]
2. Project Definition, Research Aims and Objectives <ul style="list-style-type: none"> •Identify areas of interest •Select Research Topic 	Based on chosen research field within unmanned aerial vehicles, student to provide insightful research definition, research course and research requirements	Project Defined. Project Aims and Objectives well-spelt out and concisely summarised in a two page report submission to supervisor for review. [Second Week of Project]

3. Review of Proposal Research proposal evaluated against student's ability, and potential contribution to research field	Student and supervisor meet to discuss review of proposal and identify hardware and software requirements for project.	Draft of research proposal edited based on review and final proposal submitted to department. [Third Week of Project]
4. Project Management and Tasks Identification •Algorithms, simulations, and experimental tasks.	Based on clear understanding of course of research, research objectives broken down into task-bits with deadlines for the accomplishment of each task	•Produce Gantt Chart with exceptionally clear, detailed and dated timeline for tasks completion. [Fourth Week of Project]
5. Introduction to Dissertation Report •Identify scope of UAV applications, UAV history and classification in the light of research focus	To understand scale of UAV knowledge that already exist, appreciate their applications and usefulness of research to UAV applications.	•Produce interim report on project introduction. [Fifth Week]

<p>6. Literature review</p> <ul style="list-style-type: none"> •Explore current research in autonomous UAV landing and development; •Identify methods for achieving autonomous landing of a UAV •Review available methods and develop an approach for project 	<p>Based on acquired knowledge, get understanding of current practice and methods and find a way to improve upon pre-existing results</p>	<ul style="list-style-type: none"> •Critical review of conference proceedings, journals and other literatures •Produce draft of report to include findings [Sixth and Seventh Weeks of Project]
<p>7. UAV Flight Development</p> <ul style="list-style-type: none"> •Explore and understand [27]'s work •Identify loopholes in work •Fix UAV and test flights in- 	<p>Enables knowledge of flight management system components such as gyros, accelerometers, rotors and propellers;</p> <p>Critically review flight requirements and possibly establish cause for discrepancies in flight implementation;</p>	<ul style="list-style-type: none"> •Discovered current draw of prop motors were beyond supplied battery current; [Week 7] •Discovered props on quadrotor were inaccurately calibrated leading to different supply voltages to the puller and pusher props simultaneously; this led to

laboratory		<p>vibrations and high instability during flight [Week 8]</p> <ul style="list-style-type: none"> ●Recommendations were made to the department for an AC to DC converter based on calculated current and voltage necessary to powering motors. [Week 8]
8. Explore compatible camera technology	<p>Get an overview of third-parties, off-the-shelf and compatible camera sensor boards; Based on findings recommend a scalable and modular architecture to department</p>	<ul style="list-style-type: none"> ●Based on findings, and described feature capabilities, it was thought that the CMUcam4 [28] would best suit project needs due to its UART support, arduino compatibility and most of all programmability and other added features; ●Recommendations made for CMUcam4 was procured. [Week 10].

<p>9. Integrate camera to UAV</p> <ul style="list-style-type: none"> • Explore flight with camera on-board • Explore camera board programmability from a central interface alongside the UAV 	<p>Empowers knowledge ascendance based on camera integration techniques, communication development between the CMUcam4 and UAV autopilot, camera control during flight and</p>	<ul style="list-style-type: none"> • Default frequency of frame capture by CMUcam4 upgraded through <i>SPIN®</i> programming; <i>(based on release by Kwabena Agyeman^[28] on June 30, 2012)</i> • With the arduino uno board, camera was used in test flights to acquire mock landing targets images
<p>10. Interface camera with ATmega 2560 Board</p> <ul style="list-style-type: none"> • Test 	<p>Explore different ways of interfacing the CMUcam4 to the UAV, test different methods and adopt approach that gives best performance, speed and little camera distortion during flight.</p>	<ul style="list-style-type: none"> • Using spare UART serial connection terminals on the arducopter oilpan, the CMUcam4 was interfaced with the UAV and programmed via the arduino sketch IDE ®

APPENDIX B

CMUCAM4 Automatic Paparazzi Program

```
AutomaticPaparazzi_640x480 | Arduino 1.0.1
File Edit Sketch Tools Help

AutomaticPaparazzi_640x480

#include <CMUcam4.h>
#include <CMUcom4.h>

#define RED_TOLERANCE 30
#define GREEN_TOLERANCE 30
#define BLUE_TOLERANCE 30

CMUcam4 cam;

void setup()
{
    cam.begin();
}

void loop()
{
    // Data structures.
    CMUcam4_statistics_data_t base;
    CMUcam4_statistics_data_t sample;

    // Wait 5 seconds for the scene to settle.
    cam.LEDOn(2);
    delay(5000);

    // Start "getMean" mode streaming.
    cam.LEDOn(0);
    cam.getMean();

    // Capture base line statisitcs.
    cam.getTypeSDataPacket(&base);

    do
    {
        // Continously capture image statistics.
        cam.getTypeSDataPacket(&sample);
    }
    while
    {
        // Look for any difference in means.
        (abs(base.RMean - sample.RMean) < RED_TOLERANCE) &&
        (abs(base.GMean - sample.GMean) < GREEN_TOLERANCE) &&
        (abs(base.BMean - sample.BMean) < BLUE_TOLERANCE)
    };

    // Something changed.
    cam.idleCamera();
    cam.LEDOn(5);
    delay(5000);

    // So take a picture.
    cam.dumpFrame(CMUCAM4_HR_640, CMUCAM4_VR_480);
    cam.unmountDisk();
}
```

Python Program

Navigation of Quadrotor from Initial Location to Landing Area

```
print 'Start Script'
for chan in range(1,9):
    Script.SendRC(chan,1500,False)
Script.SendRC(3,Script.GetParam('RC3_MIN'),True)

Script.Sleep(5000)

Script.SendRC(3,1000,False)
Script.SendRC(4,2000,True)
cs.messages.Clear()
Script.WaitFor('ARMING MOTORS',30000)
Script.SendRC(4,1500,True)
print 'Motors Armed!'

Script.SendRC(3,1700,True)
while cs.alt < 7:
    Script.Sleep(50)

Script.SendRC(5,2000,True) # acro

Script.SendRC(2,2000,False) # roll
Script.SendRC(3,1370,True) # throttle
while cs.pitch > 6: # Move 6m from starting position to landing target
    Script.Sleep(5)
while cs.pitch < 6: # Stop at 6m point
    Script.Sleep(5)

Script.SendRC(5,1500,False) # stabilize
Script.SendRC(2,1500,True) # level pitch

Script.Sleep(60000) # 10 sec to stabilize and acquire image
Script.SendRC(3,1300,True) # throttle back to land

thro = 1350 # will descend

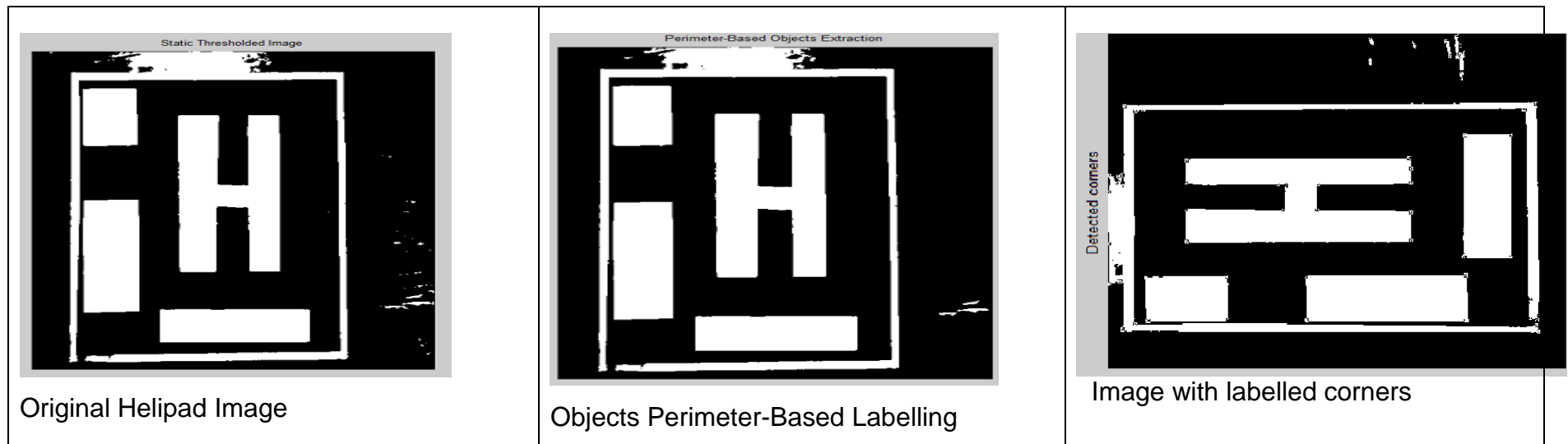
while cs.alt > 0.1:
    Script.Sleep(300)

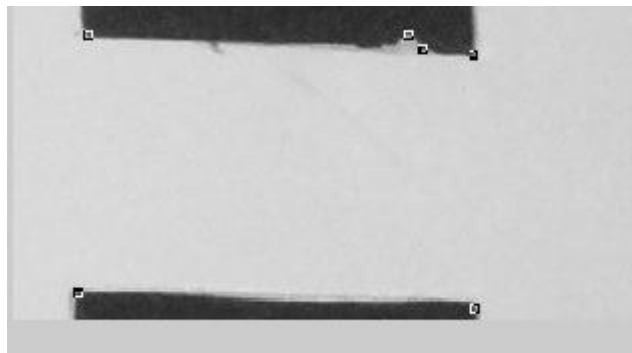
Script.SendRC(3,1000,False)
Script.SendRC(4,1000,True)
Script.WaitFor('DISARMING MOTORS',30000)
Script.SendRC(4,1500,True)

print 'Take Off, Hover over Landing Pad complete'
```

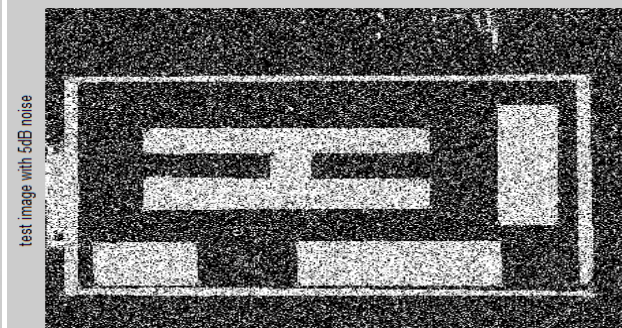
APPENDIX C

Results of Low Level Image Analysis

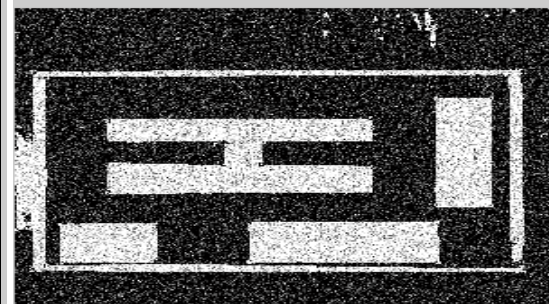




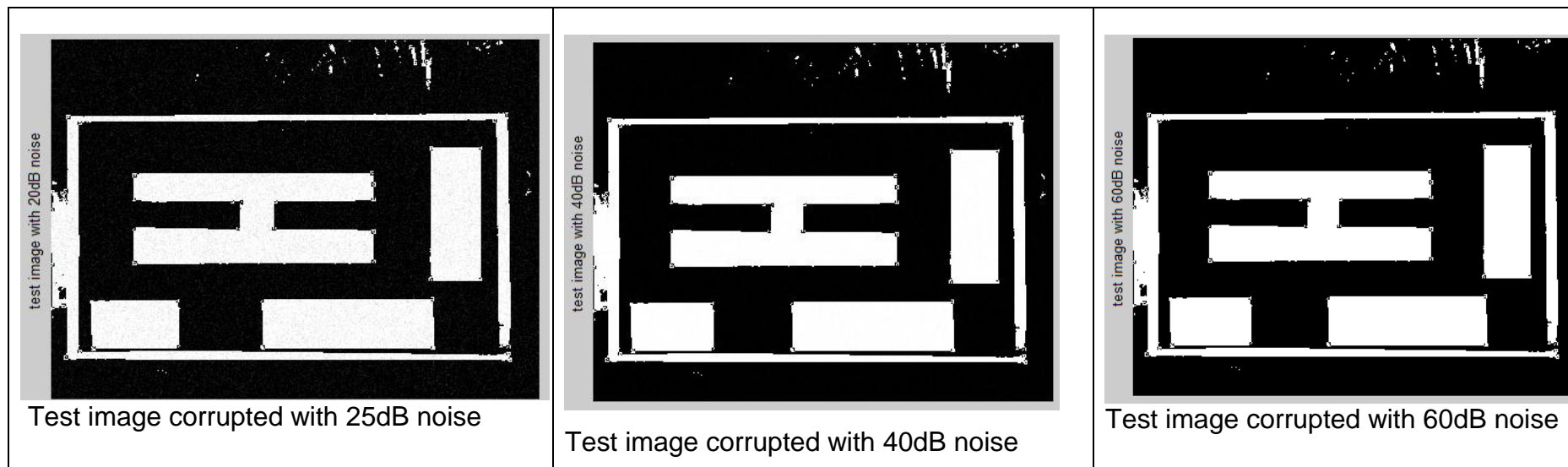
Example image with false corners



Test image corrupted with 5dB noise



Test image corrupted with 10dB noise



28. FIGURE C.1

APPENDIX D

D.1 BOLTZMANN MACHINE WORKING MECHANISMS

Numbers called “energy” are assigned to a global state of a network system, and distinct computing elements called units are used to minimize global energies such that the machine can reconstruct the remainder of an incomplete data it is presented with by deriving internal variables that gave rise to the partial example. These are used to complete data or patterns.

The energy of a global configuration is defined as,

$$E = -\sum_{i<j} w_{ij} S_i S_j + \sum_i \theta_i S_i \quad (D.1)$$

where w_{ij} is the connection strength between i and j , S_i is the state $\{0,1\}$ based on unit i , and θ_i is the threshold of unit i .

To avoid the algorithm from getting restricted in local minima (particular to a deterministic problem such as this), noise is used to escape via the decision rule:

$$\rho_k = \frac{1}{\left(1 + e^{-\Delta E_k/T}\right)} \quad (D.2)$$

where T is a temperature-like variable. A connected system of units obeying the decision rule in (D.2) will follow the Boltzmann distribution relating the energy of a state to the negative log probability of that state as follows:

$$\Delta E_i = -k_B T \ln(p_i = 0) - (-k_B T \ln(p_i = 1)) \quad (D.3)$$

where $p_i = 0$, or 1 designate the off and on states respectively and k_B is the Boltzmann constant.

Network is run by recurrently selecting a unit and setting its state according to (D.2). Probability distributions of global states converge at thermal equilibrium resulting in *simulated annealing*¹ (49).

Training is done by setting weights so that global states with highest probabilities get the lowest energies. The units of the Boltzmann Machine are divided into two sub partitions called the visible and hidden units. The visible units interact with the external environment (training) and are “*clamped*” into different labels of training data such that when testing for completion ability, the subsets of the visible units are grouped. The hidden units (or testing data) are used for explaining constraints in the category of test data not representable by “pairwise constraints among visible units” (49).

D.2 RESTRICTED BOLTZMANN MACHINES MECHANISMS

For the pattern we want to learn, the corner pair positions are first converted to binary for concreteness and computational efficiency.

The probability of specific settings of the hidden and visible units is defined as:

$$P(V, H) = \frac{\exp(-E(V, H))}{Z} \quad (D.4)$$

where Z is a partition function that normalizes P(V, H) defined as:

$$Z = \sum_{V, H} \exp\{-E(V, H)\} \quad (D.5)$$

In (43), Smolensky derived the expression relating $P(H|V)$ and $P(V|H)$ and it is presented thus:

$$P(V|H) = \prod_i P(H_i|V). \quad (i = 1, 2, 3 \dots) \quad (D.6)$$

¹ *Simulated annealing* locates a good approximation to the global minimum of a given function in a large search space.[47]

$$P(H = 1|V) = \sigma(C + V^T W) \quad (D.7)$$

where σ is the logistic sigmoid, $\sigma(x) = (1 + e^{-x})^{-1}$ and the symmetric version of (5.11) is given by

$$P(V = 1|H) = \sigma(B + H^T W^T) \quad (D.8)$$

Equations (D.7) and (D.8) have the terms $V = 1$ and $H = 1$ because the layers can either be in the off or on state and the on state has been chosen. Equation (D.8) allows the use of weights of an RBM to assign an initial feed-forward neural network with sigmoidal hidden units as inference for RBM hidden units because forward propagation can be equated with inference for RBM hidden units in a neural network.

The log likelihood assigned to a neural network for training is therefore given by,

$$l(\theta) = -\varphi(v) - \log \sum_v \exp\{-\varphi(V)\} \quad (D.9)$$

where θ represents the model parameters, $\varphi(V)$ is the free energy defined as

$$\varphi(V) = -\log \sum_H \exp\{-E(V, H)\} \quad (D.10)$$